QUANTIZATION, CALIBRATION AND PLANNING FOR EUCLIDEAN MOTIONS IN ROBOTIC SYSTEMS

by

Sipu Ruan

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland

August 2020

Abstract

The properties of Euclidean motions are fundamental in all areas of robotics research. Throughout the past several decades, investigations on some low-level tasks like parameterizing specific movements and generating effective motion plans have fostered high-level operations in an autonomous robotic system. In typical applications, before executing robot motions, a proper quantization of basic motion primitives could simplify online computations; a precise calibration of sensor readings could elevate the accuracy of the system controls. Of particular importance in the whole autonomous robotic task, a safe and efficient motion planning framework would make the whole system operate in a well-organized and effective way. All these modules encourage huge amounts of efforts in solving various fundamental problems, such as the uniformity of quantization in non-Euclidean manifolds, the calibration errors on unknown rigid transformations due to the lack of data correspondence and noise, the narrow passage and the curse of dimensionality bottlenecks in developing motion planning algorithms, etc. Therefore, the goal of this dissertation is to tackle these challenges in the topics of quantization, calibration and planning for Euclidean motions.

Thesis Committee

Primary Readers

Gregory S. Chirikjian (Advisor)
Professor
Department of Mechanical Engineering
Johns Hopkins University

Louis L. Whitcomb
Professor
Department of Mechanical Engineering
Johns Hopkins University

Noah J. Cowan
Professor
Department of Mechanical Engineering
Johns Hopkins University

Marin Kobilarov
Assistant Professor
Department of Mechanical Engineering
Johns Hopkins University

Acknowledgments

This five-year PhD journey is a special and memorable experience in my life. I would firstly express my sincere gratitude to my PhD advisor Prof. Gregory Chirikjian. He not only provides a lot of insightful and inspiring ideas as well as precious outreach opportunities that guide my research, but also offers considerable kindness in daily life. I would also thank my other thesis committee members, Prof. Louis Whitcomb, Prof. Noah Cowan and Prof. Marin Kobilarov, for their valuable evaluations and suggestions. Also, during the earlier years, their fantastic courses led me to the amazing world of robotics.

I am so lucky to meet many talented people during these years, without whom I could not achieve what I have today. Firstly, I would like to thank the members in my research lab – Robot and Protein Kinematics lab – for useful academic discussions and maintaining good personal relationships, specially including Karen Poblete, Thomas Mitchel, Hongtao Wu, Yuanfeng Han, He Chen, Weixiao Liu, Qianqiang Zhao, Amirreza Fahim Golestaneh, Qianli Ma, Jin Seob Kim, Mengdi Xu, Shengnan Lyu, Christian Wülker, Zachariah Goh, Arash Ghaani Farashahi, Kristopher Reynolds, Yuttana Itsarachalyot, Jianzhong Ding, Ting Da, Fan Yang and Ya Deng. Also, I thank other friends

and colleagues in Johns Hopkins University for their continuous supports, specifically including Xinzhi Xue, Ratan Othayoth, Mahsan Bakhtiari-Nejad, Joshua Liu, Yixuan Wu, Long Qian, Shahin Lashkari, Qiaozhi Wang, Weiqi Wang, Christie Opiekun, Farshid Alambeigi, Felix Jonathan, Shahriar Sefati and Amirhossein Farvardin. Plus, I would also thank many collaborators outside JHU, including professors and visiting students in conducting various interesting projects, who have all helped me a lot.

Last but not least, I sincerely thank my parents for their unconditional love and supports during this period. From the first day I leave them for another country till now, they always encourage me and share any happiness or hardness in the daily life. Although they do not express much about sadness when I am not with them for a long time, I can feel how hard they are during the time I am alone in the other half of the globe. I thank them for providing the best of they can to support my decisions, with which I can keep full energy to reach my goals and overcome any barriers in my entire life.

To my parents.

Table of Contents

List of Tables

List of Figures

Part I

Overview and Preliminaries

Chapter 1

Introduction

Humans are experts at planning motions. We are able to recognize the surrounding environment based on prior knowledge and quickly make decisions on which path is viable, even when the allowable space is limited. However, these common sequences of actions are always challenging to robots. Throughout the past several decades, investigators have put quite a lot of efforts on sensing the environment precisely and generating feasible motion plans for the robots. Various types of autonomous robotic systems have been developed such as self-driving cars, unmanned aerial vehicles, autonomous underwater vessels and intelligent humanoid robots. The well-organized operations within these systems require effective and efficient algorithms at the back-end, whose foundations are the studies of basic Euclidean motions using different types of mathematical approaches.

1.1 Motivations

At the macroscopic level, the world we are living on can be viewed as continuous. But in order to describe and store important features, people are using discrete symbols as representatives of this continuous space. This representation technique is called, in general, the *signal-to-symbol* transformation in classical artificial intelligence (AI) [114]. For example, human languages encode meaningful expressions in discrete letters, which can be combined into words and sentences; genetic information of all living species is composed by sequences of the four basic elements, i.e. Adenine (A), Cytosine (C), Guanine (G) and Thymine (T); viewable figures are generated by digital pixels with limited range of values, but are able to show and store the perceptional features effectively. Analogously, robot motions can also be quantized via basic motion primitives, which enables efficient encoding and decoding processes in storage and executions.

Therefore, the first main topic that this dissertation addresses is:

1. Providing an efficient framework to quantize Euclidean motions into a motion alphabet, and decoding an arbitrary motion into its closest motion primitive.

In literature, the objective is similar to the sampling theory on motions, which limits to some specific motion types such as 3D rotations. However, with the help of Lie groups theory, a discrete set of general Euclidean motions with nice group structures and uniformity properties can be generated. With this quantization framework, the arena that the robot moves can be divided into equi-spaced Voronoi-like cells, and a given motion can be simply rounded by a combination of the representatives from the sampling set.

Once the space of Euclidean motions is quantized well enough, before actually executing the robot motions, a precise sensing of such an arena is required. This needs another important building block in a robotic system — calibration. There are substantial amounts of work that investigate extrinsic sensor calibrations, most of which require the full data correspondence [145]. However, this requirement might not always be available since there might be some missing data points or the ordering of data are not always correct [1]. Therefore, the second topic is:

2. Developing a novel probabilistic algorithm, which should be robust to scrambled and noisy data streams without exact temporal correspondence.

The algorithm developed here is an extension to the recently proposed probabilistic methods by modeling the calibration expressions as probability density functions on the special Euclidean group, $\mathrm{SE}(3)$. The basic framework iteratively minimizes the errors introduced on the equations of calibration, thereby allowing larger tolerance on the noise from the data streams collected from the sensors.

Both the topics of quantization and calibration apply the *group-theoretic* approaches in robotics, such as the concept of double-coset decomposition, probabilities and convolutions on Lie groups. Though group theory was introduced into physics more than a century ago, it only recently has been widely recognized as an effective mathematical tool to describe motions and transformations in modern robotics. Both quantization and calibration can be viewed as recognizing and characterizing the environment for the robot to enable it to execute motions through the use of group theory. Therefore, the

two topics are organized in the first part of this dissertation as the perception module before developing a motion planning paradigm.

With the two topics introduced above, a motion plan can then be computed to guide the robot moving through a cluttered environment while avoiding obstacles. Therefore, the third topic of this dissertation is:

3. Developing several algorithms to efficiently plan feasible path of motions for different types of robots based on the parameterizations of the collision-free configuration space.

In this topic, the concept of *configuration space* is used based on explicit computations of *Minkowski sums* between the robot parts and environmental components. By definition, one configuration of a robot is the complete specification of the position of all points in the system, and the space of all configurations is called the configuration space (C-space) [38]. The Minkowski sums characterize the collision (or contact) space between two objects, which has been investigated intensively for decades.

Apart from discrete and faceted objects, smooth bodies with implicit and parametric surfaces are the main geometric primitives to be studied in this topic, where ellipsoids and superquadrics are the typical examples. Studies found that Minkowski sums for ellipsoids can be parameterized in *closed-form* with computational complexity only depending on one body [164]. The idea is extended for a general body, which formulates the geometric foundations of the motion planning algorithms.

As important submodules in a motion planning framework, proximity queries and containment space characterizations are introduced at first. Then,

the motion planning algorithm for complex rigid-body robots is built, which guarantees safety and runs fast. In addition, the case when the robot has higher degrees of freedom is solved by a hybrid algorithm. It is hybrid since collision-free configurations are computed in collaboration with the spirit of stochastic sampling of the robot shapes. These samples are then fed into sampling-based planner as seeds without further explicit collision detection. By this means, a motion plan through a narrow and cluttered environment can be obtained to deal with the curse of dimensionality problem in an effective way.

1.2 Organization of This Dissertation

The whole dissertation consists of four main parts, each of which has several self-contained chapters.

The chapters in the rest of the first part provides literature review and some essential basic mathematical fundamentals for this dissertation. Based on the mathematical tools and application backgrounds, the three major topics are presented in the two following major parts.

Part II introduces two applications using the group-theoretic approaches prior to robot motion planning. The first topic discusses quantization of Euclidean motions by generating an alphabet via double-coset decomposition into Voronoi-like cells. This equi-volumetric discretization on Lie groups obtains high uniformity in dividing the motion space, resulting in very efficient decoding algorithms for a specific movement sequence. The second topic deals with precise calibration methods for non-correspondent data streams

from sensor readings, via probability density approximations on the group of rigid-body motions. The novel algorithm proposed handles larger noise and covariance of the data well through validations on both simulated and physical experiments. Both of these two topics apply the group-theoretic approaches to naturally model and solve the problems as well as to simplify computations.

Part III proposes several useful and efficient frameworks in solving complex robot motion planning problems based on the parameterization of free space. Proximity and containment queries are discussed at first, via detailed computations of the contact space between robot parts and environmental components. Then, efficient motion planning algorithms are proposed using the novel closed-form characterizations of the contact space of robots, whose rigid parts are encapsulated by ellipsoids. Benchmarks with the modern sampling-based algorithms illustrate the efficiency and scalability of the proposed framework to the narrow passage and high dimensional problems, both of which are well-known challenges in the field of geometric motion planning.

At last, Part IV concludes the whole dissertation, points out some potential future directions and provides appendices for necessary derivations and proofs.

The main ideas of each chapter are summarized as follows.

- Chapter 2 reviews related work on Lie groups methods in robotics and some challenges in robot motion planning problems. It first reviews some basic concepts of a group, followed by the applications on quantizations of Euclidean motions and the solutions to the extrinsic sensor

calibration problems (particularly for the multi-robot $AXB = YCZ$ calibration settings).

- Chapter 3 reviews necessary concepts and properties of Lie groups as well as some related novel geometric tools.

- Chapter 4 develops a novel *signal-to-symbol* algorithm in quantizing robotic motions via double-coset decomposition of groups [31], [125], [158]. The study proposes algorithm to discretize the continuous motion space of a robot in both $\mathrm{SE}(2)$ and $\mathrm{SE}(3)$ configuration spaces, and to efficiently locate a specific robot pose into its nearest neighboring representative in a *coarse-to-fine* manner.

- Chapter 5 proposes a new probabilistic algorithm to solve for the $AXB = YCZ$ calibration problem [101], where the data stream has no correspondence. The algorithm iteratively mitigates the variations from the errors of the calibrating equations, and is robust to deal with sensor noises through both the simulated and physical experimental verification.

- Chapter 6 answers the collision and proximity queries between one ellipsoid and a general surface with implicit and parametric expressions in $\mathbb{R}^n$ [127]. The algorithms are based on the closed-form Minkowski sums expressed in parametric form, and solve for nonlinear optimization problems for contact and shortest distance with a specific point. This problem is an important submodule for sampling-based motion planners since these frameworks depend heavily on efficient collision detectors.

- Chapter 7 focuses on the kinematics of containment for ellipsoidal bodies

in $\mathbb{R}^n$ [124]. The idea is to explicitly characterize the free motion space for an ellipsoid being fully contained inside another large one. Both algebraic and geometric methods are proposed and verified in 2D and 3D cases. The study is closely related to developing an efficient motion planning algorithm and error space analysis in a robotic assembly task.

- Chapter 8 proposes efficient motion planning algorithms for complex robots articulated by ellipsoidal parts [126]. The core mathematical foundation is based on the closed-form Minkowski sums that efficiently parameterizes the boundary of collision-free configuration space. Algorithms for both rigidly-connected and articulated bodies are proposed and compared with sampling-based planners from the standard benchmark library. Physical experiments are then conducted to verify the efficiency in real-world settings.

- Chapter 9 concludes the whole dissertation and points out some potential future work that worth more investigations.

- Appendix A provides some detailed derivations and proofs for the geometric properties of superquadrics and ellipsoids. Section A.1 provides explicit expressions for the closed-form Minkowski sums for the case of superquadric objects. Section A.2 derives the reconstruction of superquadrics from point cloud data. Section A.3 proves the minimality of the concentric ellipsoid bounding two ellipsoids with the same center. Section A.4 computes the extreme distance a sphere can move along the semi-axis of an ellipsoid in $\mathbb{R}^n$.

- Appendix B derives the some essential expressions of probabilities on SE(3) for the $AXB = YCZ$ calibration problem. Section B.1 proves the second order approximations of the three-fold convolution. Section B.2 gives detailed derivations of the proposed iterative refinement algorithm.

1.3 Contribution Statements

This section states, explicitly, my contributions to the related publications in each topic. The summary is divided as a list of related publications, and my contributions are stated for each publication. The list follows the order of the chapters.

1. *"Wuelker, C., **Ruan, S.** and Chirikjian, G.S., 2019. Quantizing Euclidean motions via double-coset decomposition. Research, 2019, p.1608396."*
   This article is the core for Chapter 4. In the theory part, I proposed a hybrid nearest neighbor searching algorithm for an arbitrary sampling set of a Lie group. And I benchmarked all the algorithms introduced in the article with some existing methods.

2. *"Chirikjian, G.S., Mahony, R., **Ruan, S.** and Trumpf, J., 2018. Pose changes from a different point of view. Journal of Mechanisms and Robotics, 10(2)."*
   This article motivates the decomposition of SE(3) in Chapter 4 when translation and rotation parts are separated. I generated all the illustrative figures and worked out the application section on path interpolations using the proposed "pose change group".

3. *"Ma, Q., Goh, Z., **Ruan, S.** and Chirikjian, G.S., 2018. Probabilistic approaches to the $AXB = YCZ$ calibration problem in multi-robot systems. Autonomous Robots, 42(7), pp.1497-1520."*

   This article is related to Chapter 5. I proposed a new probabilistic iterative algorithm for the calibration problem using the full information of mean and covariance equations. Also, I conducted physical experiments using two NAO humanoid robots to validate the proposed algorithm, which shows supe riority in dealing with noisy data streams compared to the previous probabilistic algorithm.

4. *"**Ruan, S.**, Poblete, K.L., Li, Y., Lin, Q., Ma, Q. and Chirikjian, G.S., 2019, May. Efficient Exact Collision Detection between Ellipsoids and Superquadrics via Closed-form Minkowski Sums. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 1765-1771). IEEE."*

   This paper builds up Chapter 6. I was leading all the work in this publication. In particular, I proposed a collision detection algorithm using the closed-form Minkowski sum expression. Then I benchmarked the proposed collision checker with some existing algorithms to show its efficiency. I also added some new contents that query distance between the two bodies in this chapter.

5. *"**Ruan, S.**, Ding, J., Ma, Q. and Chirikjian, G.S., 2019. The kinematics of containment for N-dimensional ellipsoids. Journal of Mechanisms and Robotics, 11(4)."*

   This article, along with the conference version, is in Chapter 7. I was the leading author and did all the work including the extended theory and

simulations.

6. *"**Ruan, S.**, Ma, Q., Poblete, K.L., Yan, Y. and Chirikjian, G.S., 2018, December. Path planning for ellipsoidal robots and general obstacles via closed-form characterization of Minkowski operations. In International Workshop on the Algorithmic Foundations of Robotics (pp. 3-18). Springer, Cham."*
   This paper formulates the core contents of Chapter 8. I was leading all the work including developing the path planning algorithm, benchmarking with sampling-based planners and conducting physical experiments using a humanoid robot.

Chapter 2

Literature Review

This chapter reviews related work from literature for the development on Lie group theories in robotics and important issues in the robot motion planning field. Section 2.1 provides definitions of a Lie group followed by the applications on robotic motion quantization and calibration. Section 2.2 reviews the field of motion planning from several aspects: the challenge of *narrow passage* problems raised from sampling-based planning algorithms (Section 2.2.1); the geometric representations of rigid objects, especially using the ellipsoidal and superquadric models (Section 2.2.2); the computational strategies for Minkowski sums that describe the free space as a prior knowledge for a motion plan (Section 2.2.3); and the important problem of collision detection submodule in a motion planning query (Section 2.2.4).

2.1 Lie Groups in Robotics

Group theory has been used for decades [10], [17], [143], and is a popular and effective tool for solving problems related to motions in robotics community

[100], [110], [111], [132]. Group-theoretic methods as well as a more general field of abstract algebra have also been sprinkled in literature [4], [56], [63], [65], [80], [96], [106]. Specifically, group theory provides natural and convenient ways to describe complex motions, and formulates universal frameworks that have been accepted in both academic research and industry applications.

By definition, a group G with operation $\circ$ is a set that satisfies the following properties [30]

- *Closure*: if $g_1, g_2 \in G$, then $g_1 \circ g_2 \in G$;
- *Identity*: there exists a unique identity element $e \in G$, such that for $g \in G$, $e \circ g = g$;
- *Inverse*: for every $g \in G$, there exists a unique inverse, denoted by $g^{-1} \in G$, such that $g^{-1} \circ g = e$, where $e \in G$ is the identity element;
- *Associativity*: if $g_1, g_2, g_3 \in G$, then $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

Throughout the context of this dissertation and the study of robotic motions, a very important special case of group theory is of interests – *matrix Lie group* [25]. The element $g \in G$ of a matrix Lie group is an $N \times N$ matrix, with group operation $\circ$ being the matrix multiplication, and the group operations as well as inverse are analytic. Also, given a matrix Lie group, for the element $g \in G$ near identity, there is an associate Lie algebra $X \in \mathcal{G}$ such that $g(t) = \exp(tX)$, where t is near 0 and exp is the matrix exponential. Inversely, to compute the Lie algebra of a Lie group, the matrix logarithm is applied.

The set of all rigid-body motions forms a matrix Lie group $\mathrm{SE}(n)$, which has subsets of rotation groups $\mathrm{SO}(n)$ and Euclidean spaces $\mathbb{R}^n$. More precise

definition of these important and useful groups will be reviewed in the following chapter. The rest of this section reviews some related work from literature about two distinct applications of Lie groups: motion quantization and sensor calibration.

2.1.1 Applications on Motion Quantization

Discretization of Euclidean motions has attracted significant interests throughout the years [81]. In particular, the uniform sampling of rotations, either random or deterministic, has a wide range of applications including computer graphics [6], [136], protein crystallography [85], molecular physics [47], materials science [107], and robot motion planning [161], [167]. In the field of crystallography, there is a family of discrete groups that well-quantizes the motions of a protein molecule, i.e. the *crystallographic groups*. Precisely, the crystallographic group is a discrete subgroup of the Euclidean group, including not only rotations but also reflections and improper rotations. It describes the symmetry associated with the phase transition in a crystal solid [140]. In the 3D case, there are a total of 230 different groups, which are summarized and can be searched through an online engine [5], [61]. Some notable introductions to mathematical crystallography include [5], [61], [72], [156]. Because they are discrete and each has various number of elements, it is able to quotient them from their super-group and formulate a coarse representation of the Euclidean motions under different resolutions. The criteria for a good quantization method depend mainly on the level of uniformity in covering the space and the running time [105].

In Chapter 4, the proposed quantization algorithms use the crystallographic group extensively and are compared with existing methods in the case of 3D rotations. The discretization of continuous motions will generate part of an alphabet, from which the discrete words that capture the essence of a continuous motion/action can be constructed.

2.1.2 Applications on Sensor Calibration

The theory of Lie group is used commonly in the problem of robotic sensor calibration, specifically in solving the extrinsic relative transformations between the cameras and robot body frames. Three well-known types of problems are studied for decades, i.e. the $AX = XB$ [1], [42], [49], [102], [117], [135], [145], $AX = YB$ [46], [88], [89], [168] and $AXB = YCZ$ [101], [149], [157], [159]. In these problem formulations, A, B, C are rigid-body transformations extracted from the sensor readings, and X, Y, Z are the unknown transformations from the mounted sensors (i.e. ultrasound probe, camera, etc.) to the robot body frames.

For the specific case of the $AXB = YCZ$ calibration problem, which is the main objective of Chapter 5, traditional solutions assume full correspondence among the data streams from sensor measurements. Among the work in this specific problem, Degradation-Kronecker and purely nonlinear methods were presented and verified in a hybrid serial-parallel robot. The algorithm enables a simultaneous registration of the robot-world and tool-flange transformations [159]. In addition, [157] proposed a closed-form solution for recovering the rotational part of the unknowns using quaternions, which is experimented

in a dual-arm system. Recent developments of probabilistic solvers successfully have solved for the multi-robot calibration problem without the data correspondence using probabilities on Lie groups [101].

The work in this dissertation is built on top of the previous probabilistic solvers. But instead of separately solving for rotation and translation parts, the algorithm proposed here simultaneously and iteratively recovers all the unknown transformations. Compared to the previous work, which only validated in simulations, the new algorithm is further verified experimentally in a real humanoid system.

2.2 Robot Motion Planning

Over the decades, robot motion planning problems have been extensively studied, with Hundreds of remarkably successful algorithms having been proposed and implemented. In the early years, algorithms such as visibility graph [99], cell decomposition [22], potential fields [78] opened the doors of interests for geometric planning scenarios (i.e. the classical "piano mover's" problem) in low dimensional configuration space (C-space). A general motion planning problem has been proved to be PSPACE-hard [20], and a combinatorial algorithm provides complete solution but suffers from the exponential computational complexity [84]. As an alternative to explicitly describing the whole C-space, sampling-based algorithms marked another milestone in the motion planning community, which have proven to be effective for high dimensional problems with neat theoretical properties such as probabilistic completeness and asymptotic optimality.

Besides the algorithmic planning framework, the representation of objects (including rigid parts of the robot and encapsulation of the obstacles) also determines the effectiveness of finding a solution. Polyhedra are ubiquitously-used primitives due to their discrete nature and associated efficient algorithms. Bounding volumes such as boxes, spheres and a hierarchy of them are used due to their simplicity in computation and storage. In addition, ellipsoids and their high-order extensions – superquadrics are widely applied in motion planning for robotic grasping, unmanned aerial vehicle, and so on. These primitives only require a limited number of parameters to describe the shapes that can also tightly fit an object. Also, the orientation can be simply determined based on the body frame attached to the semi-axes.

2.2.1 The Challenge of Narrow Passages

One of the key factors that affects the performance of sampling-based planners is the configuration sampling strategy, and a simple one is to sample uniformly at random in the whole configuration space. But when there are narrow corridors, it might take too much time to find a valid collision-free sample.

To tackle this so-called *narrow passage* challenge, various of samplers have been studied throughout these years, most of which try to capture the local features around obstacles. For example, OBPRM [3] iteratively searches for collision-free samples from a configuration in collision by moving in different ray directions; and its more recent variants, such as UOBPRM [166], refines the sampling to be uniform around obstacles. The Gaussian sampler [19] picks two configurations with distance computed from Gaussian distributions,

and preserves the one when only one of them is collision free. The bridge test [66] tries to find a collision-free middle point between configurations that are in collision with the obstacles. OBRRT [123] provides hints for the growth direction of the tree structure by taking the advantage of the obstacle representations. Toggle PRM [44] and its variant [45] simultaneously map both free space and obstacle space, enabling an augmentation from a failed connection attempt in one space to the other.

Other methods combine the advantages of different kinds of algorithms, which also show effectiveness in tackling narrow passage problems. For example, [147] decomposes the free space into star-shaped regions by deterministic sampling, and the completeness of the algorithm is then guaranteed from the connectivities of the free space. Spark PRM [133] is basically a multi-query roadmap method, but grows a tree inside the narrow passage region to connect different parts of the roadmap on different ends of the region. More recently, a reinforcement learning method is applied to enhance the ability to explore local regions where the tree grows [150].

By generating more useful samples around important regions, these different kinds of planners elevate the original PRM or RRT to a more efficient and effective level. Some of them are implemented in the well-known Open Motion Planning Library (OMPL) [142], which provides us a standardized way to benchmark new algorithms.

The second part of this dissertation deals with the narrow passage problem in robot motion planning. Comparing to sampling-based planners in literature, the proposed algorithms try to explicitly parameterize the collision-free

configuration space so as to generate vertices. To solve for the dimensionality burden, collaborations with sampling-based planners are established then.

2.2.2 Ellipsoidal and Superquadric Primitives for Object Geometry

The efficiency of a motion planning algorithm depends heavily on the representation of the object geometry. Typically people are used to approximate an object using polytopes or bounding boxes. Apart from polytopes or surface meshes in object representations, other geometric primitives such as ellipsoids and superquadrics play an equally important role due to their continuous and convex features with simple algebraic characterizations. A 3D ellipsoid in general pose only needs 3 semi-axes lengths and 6 pose variables for a full definition; while for superquadrics, two additional variables are needed, i.e. the exponential factors that control the sharpness. With this limited number of parameters, a large range of shapes can be represented. Therefore in many recent applications, ellipsoids or superquadrics are good candidates to encapsulate objects [95], [109], [112], [131], [148].

Algorithms related to ellipsoids are studied extensively for efficient calculations [120], many of which are implemented and integrated in toolboxes [83]. Minimum volume enclosing ellipsoid (MVEE) is a widely-applied problem, which is characterized as a convex optimization problem and can be solved efficiently [144]. A series of papers have studied algebraic separation conditions for two ellipsoids and provide very efficient algorithms to detect collisions in both static and dynamic cases [36], [73], [151], [152]. The core idea is to first formulate the characteristic polynomial between two ellipsoids and

count for the number of sign changes of its roots. The separation status is then indicated. In addition, efficiently calculating distance between two ellipsoids is also attractive in both theory and real applications [69], [122].

Superquadrics can be seen as a high-order extension of ellipsoids, with the two additional exponents determining the sharpness and convexity [12]. They are able to represent a wide range of geometries such as cubes, cylinders, octahedra and ellipsoids. Applying the least squares optimization technique, point cloud data can be segmented and fitted by unions of superquadrics [71], [118], [137]. Proximity queries and contact detection are useful applications of this geometric model [24], [121], which are widely studied in the field like haptics [108], motion planning [8], [79], [112], manipulation [2] and grasping [148], [160].

To investigate the motion planning problems in this dissertation, rigid parts of a robot are encapsulated by ellipsoids and environmental features are enclosed by superquadrics. Apart from the nice properties discussed in literature, this dissertation further extends another important operation for these geometric primitives — Minkowski sums.

2.2.3 Computations of Minkowski Sums

The Minkowski sum is ubiquitous in many fields such as computational geometry [14], [129] and robot motion planning [84], [92], [98]. Despite its straightforward definition, computing an exact boundary of Minkowski sum between two polytopes can be computational very expensive and has attracted extensive attentions for decades. Taking two general non-convex polytopes

in $\mathbb{R}^3$ as an example, the computational complexity of their Minkowski sums can be as high as $O(m^3n^3)$, where m and n are the complexities of the two polytopes respectively. Therefore, many efficient methods (either exact or approximated) decompose the general polytopes into convex components [60], [146], since the Minkowski sums between two convex polytopes can achieve $O(mn)$ complexity [51].

In addition, point-based methods avoid using convex decomposition and computing the union of Minkowski sums, which are expensive steps. And this set of algorithms improves the performance through approximations and are easy to implement [11], [91], [119]. The major advantages are the ease of generating points than meshes, and the possibility of parallelisms [93]. But the local properties cannot be expressed by individual points themselves.

Another family applies convolution between two bodies, with the fact that Minkowski sum between two solid bodies is the support of the convolution of their indicator functions [30], [54], [59], [76]. In particular, [54] defines the negative shape and slope diagram representation, which unifies both 2D and 3D continuous objects, convex and non convex objects, and Minkowski addition and decomposition operations. A cubical Gaussian map is then used as a dual representation of 3D polytopes, which implements exact, complete, robust and efficient algorithms for Minkowski sums, collision detection and proximity queries computations [50]. Moreover, a simple approximated algorithm is proposed to avoid computing 3D rearrangement and winding numbers, and reduces the trimming issue that many convolution-based methods might face [90].

The exact complexity bounds of Minkowski sums are rigorously analyzed in [51], [52]. And the state-of-the-art implementations of both 2D and 3D Minkowski sums are available in the "Computational Geometric Algorithms Library (CGAL)" [48].

In this dissertation, the Minkowski sum between an ellipsoid and a general convex differentiable surface is computed in closed-form, which extends the previous work in the case of two ellipsoids [164]. And with this nice closed-form expression, collision detection [127] and proximity queries (Chapter 6), containment analysis [124] (Chapter 7) and motion planning [126] (Chapter 8) are solved accordingly by using algebraic and geometric tools.

2.2.4 Collision Detection as a Submodule in Sampling-based Motion Planning

An essential module in building an effective sampling-based motion planner is to perform explicit collision detections between the robot and environment. There are hundreds of well known algorithms for collision detection. Methods based on Bounding Volume Hierarchy (BVH) [58] use primitive shapes such as spheres, axis-aligned bounding boxes (AABB) or oriented bounding boxes (OBB) [57] to encapsulate polytopes. These methods have been proven to accelerate the collision detection by doing penetration test by hierarchically dividing the objects into simple geometric primitives [74]. In [109], an approach based on distance between superquadrics are defined.

Lin-Canny closest feature tracking algorithm computes the shortest distance between two polyhedra [94], which is the basis of several algorithms

for collision checking, including [39] and [68]. One of the applications of this algorithm is to check collisions for the objects that move at constant speeds. One of the drawbacks of this algorithm is the large quantity of the surface information stored in memory. The Gilbert-Johnson-Keerthi (GJK) algorithm [55], based on Minkowski formulations between two convex polytopes, does not use extensive quantities of memory and is comparably faster than the previous method. GJK does not calculate the whole Minkowski sums, but iteratively generates *simplex*. This method is guaranteed to converge after several iterations, but is still subject to the complexity of the shapes. BVH, OBB, AABB with GJK are implemented in the Flexible Collision Library (FCL) [116], which is widely used in the robotics community for collision detection purposes.

Collision detection between ellipsoids can be computed using algebraic methods [36], [37], [152]. The Algebraic Separation Conditions (ASC) are based on the characteristic polynomial equations. According to the sign and real values of the roots, it is possible to determine whether two ellipsoids are separated, touching in a point or in collision. In [97], a method similar to ASC was presented by numerically solving nonlinear equations using Newton-Raphson method with Jacobian matrices analytically calculated. This method does not rely on polyhedron-based geometries and can be extended to other shapes, but it has the inconvenience of having Jacobian singularities. In [154], a method based on the normal vectors was proposed. It formulates collision checking as a 2-dimensional unconstrained optimization problem, which requires less iterations to converge. However, it is only applicable when the

superellipsoids are expressed as a collection of smooth convex particles and an explicit relationship between the surface points and the surface normals have to be provided. Another method applied to superquadrics was introduced in [121], which based on the implicit equation of the evaluated surfaces. The contact query is expressed as a convex nonlinear constrained optimization problem. Optimization-based proximity query for implicit surfaces was proposed [21], which guaranteed to converge in polynomial time with respect to the constraints.

In Chapter 6, the collision detection and proximity queries between an ellipsoid and a general surface are studied. The major objectives are similar to the literature reviewed above. But the mathematical basics are different. In particular, the proposed algorithms are based on the closed-form Minkowski sums expressions. Most of the algorithms reviewed above directly search for closest points in the two bodies simultaneously. As a comparison to these existing methods, the dimension of variable space can be reduced to half when the Minkowski sums are expressed in closed-form.

Chapter 3

Mathematical Preliminaries

This chapter introduces several useful mathematical tools in characterizing Euclidean motions. It starts from some important properties in group theory, such as cosets, double-cosets and fundamental domains, followed by explicit derivations of three typical groups that will be extensively used in this dissertation — rotation group, Euclidean motion group and crystallographic groups. The second section focuses on some novel methods in geometry, i.e. the *closed-form Minkowski sums* between an ellipsoid and a general convex differentiable surface in $\mathbb{R}^n$; the efficient computations of a *minimum volume concentric ellipsoid* that enclosed two ellipsoids with the same center; and the concept of *kinematics of containment* that characterizes the allowable motion space for an object being fully contained by another.

3.1 Group Theory

This section reviews some important properties of groups, such as cosets, double-cosets and fundamental domains. The major contents are essential for

the double-coset decompositions of a group. Useful groups are also introduced as specific examples that are used throughout this dissertation.

3.1.1 Essential Properties of Groups

3.1.1.1 Coset and Double-coset Decompositions

A subgroup is a subset of a group, i.e. $H \subseteq G$, which is also a group and closed under the group operation of G. Then, a left/right coset is defined as follows. Suppose G is a group and H is its subgroup, denoted by $H \leq G$ (if $H \neq G$, it is then called a *proper subgroup*, denoted by $H < G$). For an element $g \in G$, the corresponding *left coset* and *right coset* are defined as

$$gH := \{g \circ h \,|\, h \in H\} \text{ and } Hg := \{h \circ g \,|\, h \in H\}, \tag{3.1}$$

respectively. A proper subgroup $N < G$ is *normal* if it is conjugated to itself, i.e. $gNg^{-1} = N$ for all $g \in G$, denoted by $N \triangleleft G$ [1].

The set of all cosets is called *coset space*. In general, the left-coset space and right-coset space, i.e.

$$G/H := \{gH \,|\, g \in G\} \text{ and } H\backslash G := \{Hg \,|\, g \in G\} \tag{3.2}$$

are different from each other. They are equal when $H \triangleleft G$, in which case $\forall g \in G,\ gH = Hg$, then $G/H = H\backslash G := \frac{H}{G}$ is a *quotient group*.

Any group can be divided into disjoint left (or right) cosets, and the number of cosets is related to the number of elements in the group and subgroup by

[1] Note that, for the sake of simplicity, the further discussions are all with respect to a proper subgroup. Therefore, the terms "subgroup" and "proper subgroup" will not be distinguished for the rest of the contexts.

the *Lagrange's theorem* [17]

$$|G/H| = |H\backslash G| = |G|/|H|. \tag{3.3}$$

Although this theorem holds for finite groups where the number of elements is countable, in general, it is also valid for Lie groups, in which case $|\cdot|$ denotes the volume.

Given two subgroups $H, K < G$, for any $g \in G$, the *double coset* is defined as

$$HgK := \{h \circ g \circ k \,|\, h \in H, k \in K\}, \tag{3.4}$$

and any $g' \in HgK$ (including $g' = g$) is a *representative* of the double coset. The corresponding *double-coset space* is then defined as

$$H\backslash G/K := \{HgK \,|\, g \in G\}. \tag{3.5}$$

3.1.1.2 Fundamental Domains

For any coset space, exactly one element per coset can be selected to form a special subset of the group called *fundamental domain*. The following will discuss the definitions and constructions of fundamental domains of coset and double-coset spaces.

Given a left-coset decomposition with respect to a subgroup H, a non-unique fundamental domain can be defined as

$$F_{G/H} \subset G\,, \tag{3.6}$$

which consists of exactly one element per left coset. A group can be partitioned

into disjoint cosets as

$$G = \bigcup_{g \in F_{G/H}} gH \,. \tag{3.7}$$

Then the group G can be reconstructed by shifting the fundamental domain by elements in H, i.e.

$$G = \bigcup_{h \in H} F_{G/H} h. \tag{3.8}$$

The right-coset case can also be defined analogously [33]. In general, such a fundamental domain is a set instead of a group. But when $H \triangleleft G$, the fundamental domain is a group with respect to the original group operation modulo H, which is isomorphic to G/H.

Similarly, given two subgroups $H, K < G$, the corresponding double-coset decomposition is

$$G = \bigcup_{g \in F_{H\backslash G/K}} HgK, \tag{3.9}$$

and G can be further reconstructed by the shifted version of the fundamental domain $F_{H\backslash G/K}$ as

$$G = \bigcup_{h \in H} \bigcup_{k \in K} h F_{H\backslash G/K} k. \tag{3.10}$$

When G is a Lie group, and H, K are *discrete* subgroups, then the fundamental domains $F_{G/H}$ and $F_{H\backslash G/K}$ have the same dimensionality with G, but smaller volumes. In this case, the fundamental domains can be constructed as Voronoi-like cells. This construction holds since G is a smooth manifold, where a proper distance metric $\rho(\cdot, \cdot)$ exists. The fundamental domains for a

Lie group can be defined as

$$\begin{aligned} &\text{for left coset:} \quad F^{\circ}_{G/H} := \{g \in G \,|\, \rho(e,g) < \rho(e,gh),\, \forall\, h \in H - \{e\}\}, \\ &\text{for right coset:} \quad F^{\circ}_{H\backslash G} := \{g \in G \,|\, \rho(e,g) < \rho(e,hg),\, \forall\, h \in H - \{e\}\}, \end{aligned} \tag{3.11}$$

where e is the identity element of G and "$H - \{e\}$" denotes the group H without identity element. For the case of double-coset, when $H \cap K = \{e\}$,

$$F^{\circ}_{H\backslash G/K} := \{g \in G \,|\, \rho(e,g) < \rho(e,hgk),\, \forall\, (h,k) \in H \times K - \{(e,e)\}\}. \tag{3.12}$$

The following subsections will review some concrete examples of typical groups that will be used throughout this dissertation, including the group of pure rotations and Euclidean motions, as well as finite groups [4], [56], [63].

3.1.2 The Group of 3D Rotations

3.1.2.1 Definitions

The 3D *special orthogonal group* that characterizes proper rotations in $\mathbb{R}^3$ is denoted as $\mathrm{SO}(3)$. By definition,

$$\mathrm{SO}(3) := \{R \in \mathbb{R}^{3\times 3} \,|\, R^{\top}R = \mathbb{I},\, \det R = 1\}, \tag{3.13}$$

where $\mathbb{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix. Here the group operation is simply matrix multiplication. And it is not hard to prove that this is a group since:

- $\mathbb{I}$ is the group identity;
- Given any two $R_1, R_2 \in \mathrm{SO}(3)$, the matrix product $R_1R_2 \in \mathrm{SO}(3)$;
- $R^{-1} = R^{\top}$ is the inverse of $R \in \mathrm{SO}(3)$; and

- Given any three elements $R_1, R_2, R_3 \in \mathrm{SO}(3)$, $(R_1R_2)R_3 = R_1(R_2R_3)$ because of the associativity of matrix multiplications.

The associated Lie algebra $\mathfrak{so}(3)$, which corresponds to infinitesimal rotations, are *skew-symmetric matrices*, i.e.

$$\Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \tag{3.14}$$

and can be transferred into the rotation matrices by matrix exponential as

$$R(\boldsymbol{\omega}) = \exp(\Omega) = \mathbb{I} + \frac{\sin\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}\Omega + \frac{1-\cos\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|^2}\Omega^2, \tag{3.15}$$

where $\Omega^\vee := \boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^\top$ defines the vector operation of skew-symmetric matrices. The opposite *hat* operation gives $\widehat{\boldsymbol{\omega}} := \Omega$.

The parameters ω_1, ω_2, and ω_3 can be thought of as Cartesian coordinates in the Lie algebra $\mathfrak{so}(3)$ (or *exponential coordinates*), and are restricted to the range $\|\boldsymbol{\omega}\|_2 \leq \pi$. They can be used to parameterize $\mathrm{SO}(3)$ through the exponential map. When $\|\boldsymbol{\omega}\|_2 = \pi$, the point is at the boundary, in which case $\boldsymbol{\omega}$ and $-\boldsymbol{\omega}$ describe the same rotation. Therefore the geometric model for $\mathrm{SO}(3)$ is an open solid ball of radius π in Euclidean space, with antipodal points identified being equivalent.

The inverse operation that maps from Lie group elements to their corresponding Lie algebra is the *matrix logarithm*. It degenerates when the rotation angle $\theta := \|\boldsymbol{\omega}\| = \pi$. By restricting the discussion to the case when $\theta < \pi$, the logarithm is uniquely defined on a subset of $\mathrm{SO}(3)$. A proper distance metric

can be defined as

$$\rho(R_1, R_2) := \left\| \log^{\vee}(R_1^{\top} R_2) \right\| \tag{3.16}$$

when $R_1^{\top} R_2$ is not a rotation by π, otherwise $\rho(R_1, R_2) = \pi$. Such a metric is *bi-invariant*, in the sense that

$$\rho(R_1, R_2) = \rho(RR_1, RR_2) = \rho(R_1 R, R_2 R), \tag{3.17}$$

where $R \in \mathrm{SO}(3)$. Using this particular metric, it is possible to construct Voronoi cells in $\mathrm{SO}(3)$ for fundamental domains of the single coset space (i.e. $F_{H\backslash \mathrm{SO}(3)}$ and $F_{\mathrm{SO}(3)/H}$) and the double-coset space (i.e. $F_{H\backslash \mathrm{SO}(3)/K}$), because Eqs. (3.11) and (3.12) become

$$F^{\circ}_{\mathrm{SO}(3)/H} = F^{\circ}_{H\backslash \mathrm{SO}(3)} = \{R \in \mathrm{SO}(3) \,|\, \rho(R, \mathbb{I}) < \rho(R, h),\ \forall\, h \in H - \{\mathbb{I}\}\} \tag{3.18}$$

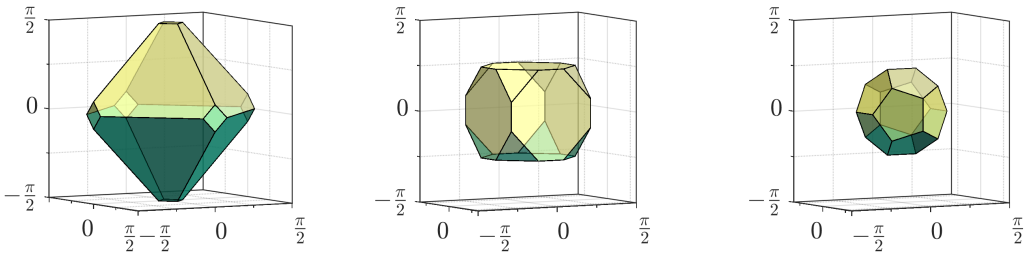
and

$$F^{\circ}_{H\backslash \mathrm{SO}(3)/K} = \{R \in \mathrm{SO}(3) \,|\, \rho(R, \mathbb{I}) < \rho(R, hk),\ \forall\, (h, k) \in H \times K - \{(\mathbb{I}, \mathbb{I})\}\}, \tag{3.19}$$

respectively.

3.1.2.2 Concrete Examples of Coset/Double-coset Decompositions

Of particular interests in this dissertation are the cases where H is one of the finite groups of rotational symmetries of the Platonic solids, as shown in Fig. 3.1 [161]. Here, the fundamental domains $F_{H\backslash \mathrm{SO}(3)}$ are depicted in exponential coordinates in $\mathfrak{so}(3)$. Note that this is a conceptual plot, since actually the edges and faces of these Voronoi cells are slightly bent inward



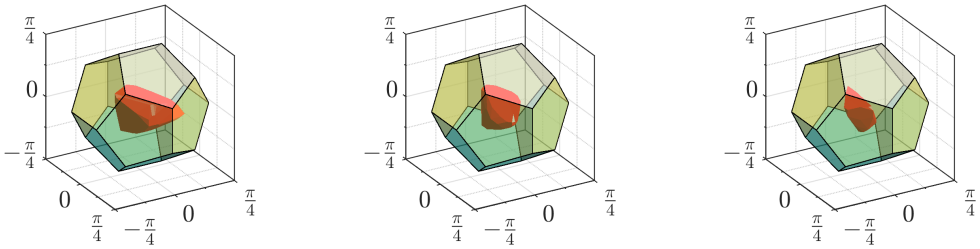
(a) Tetrahedral group **(b)** Octahedral group **(c)** Icosahedral group

Figure 3.1: Fundamental domain $F_{H\backslash \mathrm{SO}(3)}$ with H being the finite groups of rotational symmetries, constructed as Voronoi cells and viewed in exponential coordinates (figures redrawn from [158], [163]).

into the identity [34]. The cardinality (number of elements) $|H|$ in the group is 12 for the group of tetrahedral, 24 for the group of octahedral, and 60 for the group of icosahedral rotational symmetries. By the left (or right) action of H on the respective fundamental domain, it is possible to (almost completely) cover $\mathrm{SO}(3)$.

Assume H is the icosahedral symmetry group, then $F_{H\backslash \mathrm{SO}(3)}$ can be viewed as a dodecahedral cell centered at the origin of the Lie algebra $\mathfrak{so}(3)$ (Fig. 3.1c). If a second subgroup K is chosen as a conjugated group of the tetrahedral, octahedral, or icosahedral symmetries, i.e. $K := gPg^{-1}$, where P is the group of rotational symmetries of the respective Platonic solid and $g \in \mathrm{SO}(3)$ is chosen such that $H \cap K = \{\mathbb{I}\}$, then the Voronoi cell $F_{H\backslash \mathrm{SO}(3)/K}$ takes a shape as shown in Fig. 3.2.

On the other hand, if choosing $K := H$, then $F_{H\backslash \mathrm{SO}(3)/H}$ is no longer a Voronoi cell, but can be chosen as an irregular tetrahedron (the red inner region in Fig. 3.3), yielding a subdivision of the dodecahedral cell $F_{H\backslash \mathrm{SO}(3)}$ by conjugation of the tetrahedron with the elements in H.



(a) K from tetrahedral group **(b)** K from octahedral group **(c)** K from icosahedral group

Figure 3.2: Center Voronoi cell $F_{H\backslash \mathrm{SO}(3)}$ in coset space (yellow outer region) with H as the group of icosahedral symmetries, and center Voronoi cell $F_{H\backslash \mathrm{SO}(3)/K}$ in double-coset space (red inner region) with H as the group of icosahedral symmetries for all cases and K as a conjugated group of rotational symmetries (figures redrawn from [158], [163]).

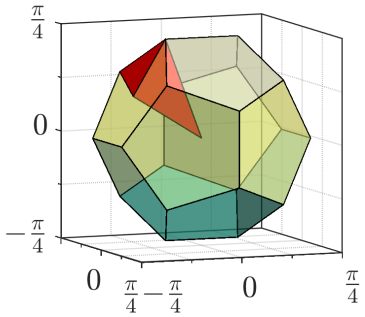


Figure 3.3: Dodecahedral cell $F_{H\backslash \mathrm{SO}(3)}$ (yellow outer region) and tetrahedral wedge $F_{H\backslash \mathrm{SO}(3)/H}$ (red inner region), with H as the group of icosahedral symmetry. The dodecahedral cell can be decomposed into 60 identical tetrahedral wedges, with five packed on each pentagonal face (figures redrawn from [158], [163]).

When choosing H to be icosahedral and $K = H$ or $K = gHg^{-1}$, the group $\mathrm{SO}(3)$ can be divided into 3600 pieces of equal size. The center of each piece can be written in a unique way as $R_{ij} = h_i k_j$, where $(h_i, k_j) \in H \times K$ with $i, j \in \{1, 2, ..., 60\}$. Using this representation, any one of the 3600 points R_{ij} corresponds to a two-letter word (h_i, k_j).

3.1.3 The Group of 3D Euclidean Motions

Let $g = (R, \boldsymbol{t})$ denote a rigid-body motion relative to a reference frame fixed in space, where $R \in \mathrm{SO}(3)$ is a rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ is a translation vector. The set of all such motions forms a 6 dimensional Lie group, the *special Euclidean group* $\mathrm{SE}(3)$.

3.1.3.1 Definitions

$\mathrm{SE}(3)$ is a group because:

- The composition operation

$$g_1 \circ g_2 \;=\; (R_1, \boldsymbol{t}_1) \circ (R_2, \boldsymbol{t}_2) := (R_1 R_2, R_1 \boldsymbol{t}_2 + \boldsymbol{t}_1) \tag{3.20}$$

  satisfies the properties of both closure and associativity;

- the identity exists and is simply $e = (\mathbb{I}, \mathbf{0})$; and

- the inverse of g is $g^{-1} = (R^\top, -R^\top \boldsymbol{t})$.

The group operation is the same as multiplying homogeneous transformation matrices, i.e.

$$H(g_1 \circ g_2) \;=\; H(g_1) H(g_2), \text{ where } H(g) \;:=\; \begin{bmatrix} R & \boldsymbol{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{3.21}$$

For the Lie algebra of $SE(3)$, denoted as $\mathfrak{se}(3)$, the coordinates can be obtained as $\boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6]^\top \in \mathbb{R}^6$, and the "hat" operation can be

defined as

$$\widehat{\xi} = \begin{bmatrix} 0 & -\xi_3 & \xi_2 & \xi_4 \\ \xi_3 & 0 & -\xi_1 & \xi_5 \\ -\xi_2 & \xi_1 & 0 & \xi_6 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(3)\,. \tag{3.22}$$

The matrix exponential transfers the Lie algebra elements to their corresponding Lie group elements, which is the same as the rotation group.

In general, for an n-dimensional group, $\mathrm{SE}(n)$ is an example of a *semi-direct product* which combines the two groups $\mathbb{R}^n$ and $\mathrm{SO}(n)$ into the new group

$$\mathrm{SE}(n) := \mathrm{SO}(n) \ltimes \mathbb{R}^n. \tag{3.23}$$

The underlying set of this group is the Cartesian product $\mathrm{SO}(n) \times \mathbb{R}^n$, but the symbol $\ltimes$ reflects the fact that the group operation is not simply $(R_1, \boldsymbol{t}_1) \circ (R_2, \boldsymbol{t}_2) = (R_1 R_2, \boldsymbol{t}_1 + \boldsymbol{t}_2)$, which is also a group (called the *direct product*), but does not reflect the way that rigid-body motions work [31].

3.1.3.2 Adjoint for $\mathbf{SE}(3)$

The adjoint matrix for $\mathrm{SE}(3)$ is useful in computing the propagation of probabilities for the calibration problem in Chapter 5. It is defined as [25]

$$Ad(g) = \begin{bmatrix} R & \mathbb{O}_3 \\ \widehat{\boldsymbol{t}}R & R \end{bmatrix} \in \mathbb{R}^{6\times 6}\,. \tag{3.24}$$

If $\widehat{\boldsymbol{\xi}} \in \mathbb{R}^6$ is partitioned as $\widehat{\boldsymbol{\xi}} = [\boldsymbol{\omega}^\top, \boldsymbol{v}^\top]^\top$, then the adjoint matrix for $\mathfrak{se}(3)$ is defined as

$$ad(\widehat{\boldsymbol{\xi}}) = \begin{bmatrix} \widehat{\boldsymbol{\omega}} & \mathbb{O}_3 \\ \widehat{\boldsymbol{v}} & \widehat{\boldsymbol{\omega}} \end{bmatrix} \in \mathbb{R}^{6\times 6}\,. \tag{3.25}$$

The relationship between $ad(\widehat{\boldsymbol{\xi}})$ and $Ad(g)$ is

$$\exp\left(ad(\widehat{\boldsymbol{\xi}})\right) = Ad\left(\exp(\widehat{\boldsymbol{\xi}})\right) . \tag{3.26}$$

The inverse and product identities of adjoint matrix of $\mathrm{SE}(3)$ are useful, which are shown as follows:

$$Ad(g^{-1}) = Ad^{-1}(g) = \begin{bmatrix} R^\top & \mathbb{O} \\ -\widehat{R^\top \boldsymbol{t}} R^\top & R^\top \end{bmatrix} \tag{3.27}$$

and

$$Ad(g_1)Ad(g_2) = Ad(g_1 g_2) . \tag{3.28}$$

3.1.3.3 Probability Densities on $\mathbf{SE}(3)$

There is a unique and correct way to define integration on $\mathrm{SE}(3)$ called the *Haar measure*, denoted by dg. In particular, if $R \in \mathrm{SO}(3)$ is expressed in ZXZ Euler angles, i.e (α, β, γ), then the Haar measure can be written as

$$dg = \sin\beta \, d\alpha \, d\beta \, d\gamma \, dt_1 \, dt_2 \, dt_3 .$$

Using this measure, it is natural to define the convolution of two well-behaved functions of group elements as [25]

$$(f_1 * f_2)(g) \;:=\; \int_{SE(3)} f_1(h) f_2(h^{-1}g) \, dh , \tag{3.29}$$

where $g, h \in \mathrm{SE}(3)$. In general, such a convolution on $\mathrm{SE}(3)$ is <u>not</u> commutative in the sense that

$$(f_1 * f_2)(g) \;\neq\; (f_2 * f_1)(g) . \tag{3.30}$$

A Dirac delta function on $SE(3)$, i.e. $\delta(g)$, is defined as an infinite spike at identity and zero elsewhere:

$$\delta(g) = \begin{cases} +\infty \text{ , if } g = \mathbb{I}_4 \\ \\ 0 \text{ , if } g \neq \mathbb{I}_4 \end{cases} \tag{3.31}$$

for any $g \in \mathrm{SE}(3)$. Given $k \in \mathrm{SE}(3)$, the corresponding shifted version of the Dirac delta function can be defined as $\delta_k(g) = \delta(k^{-1}g)$. Another property is that

$$\int_{SE(3)} \delta(g) dg = 1 \,. \tag{3.32}$$

And the convolution between a function and the Dirac delta has the nice property

$$(f * \delta)(g) \;=\; \int_{SE(3)} f(h)\delta(h^{-1}g)\, dg = f(g) \,. \tag{3.33}$$

The probability density function $f(H)$ on $\mathrm{SE}(3)$ is represented by its mean M and covariance Σ, which satisfies [25], [153]

$$\begin{aligned} &\int_{\mathrm{SE}(3)} \log(M^{-1}g) f(g)\, dg = \mathbb{O}_4 \text{ , and} \\ &\Sigma = \int_{SE(3)} \log^{\vee}(M^{-1}g)[\log^{\vee}(M^{-1}g)]^{\top} f(g)\, dg \,. \end{aligned} \tag{3.34}$$

For a sequence of N sampled elements $\{g_i\}$ from $\mathrm{SE}(3)$, the corresponding discrete versions of the mean and covariance are defined as

$$\begin{aligned} &\sum_{i=1}^{N} \log(\widetilde{M}^{-1} g_i) = \mathbb{O}_4 \text{ , and} \\ &\widetilde{\Sigma} = \sum_{i=1}^{N} \log^{\vee}(\widetilde{M}^{-1} g_i)[\log^{\vee}(\widetilde{M}^{-1} g_i)]^{\top} \,. \end{aligned} \tag{3.35}$$

In practice, the mean $\widetilde{M}$ can be solved in an iterative manner [153].

Given two probability density functions, i.e. f_1 and f_2, on SE(3), the sampled mean and covariance of their convolution $(f_1 * f_2)(g)$ can be approximated to the first order as [153]

$$\begin{aligned} M_{1*2} &= M_1 M_2 \,, \text{ and} \\ \Sigma_{1*2} &= Ad(M_2^{-1})\Sigma_1 Ad^\top(M_2^{-1}) + \Sigma_2 + F(A;B)\,, \end{aligned} \tag{3.36}$$

where

$$\begin{aligned} F(A,B) = \frac{1}{4}\sum_{i,j=1}^{6} ad(E_i)\, B\, ad(E_j)^\top A_{ij} \\ + \frac{1}{12}\left\{ \left[\sum_{i,j=1}^{6} A'_{ij}\right] B + B^\top \left[\sum_{i,j=1}^{6} A'_{ij}\right]^\top \right\} \\ + \frac{1}{12}\left\{ \left[\sum_{i,j=1}^{6} B'_{ij}\right] A + A^\top \left[\sum_{i,j=1}^{6} B'_{ij}\right]^\top \right\}, \end{aligned} \tag{3.37}$$

and $A = Ad(M_2^{-1})\,\Sigma_1\, Ad^\top(M_2^{-1})$, $B = \Sigma_2$, $A'_{ij} = ad(E_i)\, ad(E_j)\, A_{ij}$, $B'_{ij} = ad(E_i)\, ad(E_j)\, B_{ij}$, with E_i denoting i^{th} unit basis vector in $\mathfrak{se}(3)$.

Note that the function $F(\cdot;\cdot)$ is bi-linear in the sense that

$$\begin{aligned} F(\alpha\Sigma_1 + \alpha'\Sigma'_1\,;\,\Sigma_2) &= \alpha F(\Sigma_1\,,\,\Sigma_2) + \alpha' F(\Sigma'_1\,,\,\Sigma_2)\,, \text{ and} \\ F(\Sigma_1\,;\,\beta\Sigma_2 + \beta'\Sigma'_2) &= \beta F(\Sigma_1\,;\,\Sigma_2) + \beta' F(\Sigma_1\,;\,\Sigma'_2)\,, \end{aligned} \tag{3.38}$$

for any constants $\alpha, \alpha', \beta, \beta'$ and any symmetric matrices Σ_1, Σ'_1, Σ_2, Σ'_2. In particular, if one of these constants is zero, then

$$F(\mathbb{O}\,;\,\Sigma_2) = F(\Sigma_1\,;\,\mathbb{O}) = \mathbb{O}. \tag{3.39}$$

Also, if both $f_1(g)$ and $f_2(g)$ are "highly-focused" in the sense that $\|\Sigma_1\| \ll 1$ and $\|\Sigma_2\| \ll 1$, then $F(\Sigma_1\,;\,\Sigma_2) \approx \mathbb{O}$.

For the three-fold convolution, the mean and covariance can be written as:

$$M_{1*2*3} = M_1 M_2 M_3\,,$$

and

$$\begin{aligned}\Sigma_{(1*2)*3} = {} & Ad^{-1}(M_2M_3)\,\Sigma_1\,Ad^{-\top}(M_2M_3) + Ad^{-1}(M_3)\,\Sigma_2\,Ad^{-\top}(M_3) \\ & + Ad^{-1}(M_3)F\left(Ad^{-1}(M_2)\,\Sigma_1\,Ad^{-\top}(M_2)\,;\,\Sigma_2\right)Ad^{-\top}(M_3) + \Sigma_3 \\ & + F\left(Ad^{-1}(M_2M_3)\,\Sigma_1\,Ad^{-\top}(M_2M_3)\,;\,\Sigma_3\right) \\ & + F\left(Ad^{-1}(M_3)\,\Sigma_2\,Ad^{-\top}(M_3)\,;\,\Sigma_3\right) \\ & + F\left(Ad^{-1}(M_3)F\left(Ad^{-1}(M_2)\,\Sigma_1\,Ad^{-\top}(M_2)\,;\,\Sigma_2\right)Ad^{-\top}(M_3)\,;\,\Sigma_3\right)\,.\end{aligned} \tag{3.40}$$

By the properties of $F(\cdot\,;\,\cdot)$, if either $\Sigma_1 = \Sigma_2 = \mathbb{O}$ or $\Sigma_2 = \Sigma_3 = \mathbb{O}$, then all the $F(\cdot\,;\,\cdot)$ terms in the covariance Σ_{1*2*3} vanish (please see Appendix B.1 for detailed derivation). This will be useful in solving the $AXB = YCZ$ calibration problem.

In addition, if the mean and covariance are M and Σ for a PDF $f(g)$, then the mean and covariance for $f(g^{-1})$ are M^{-1} and $Ad(M)\,\Sigma\,Ad^{\top}(M)$ respectively. This provides a simple way to calculate the mean and covariance of $f(g^{-1})$, which is very useful due to the frequent calculations of PDFs on the inverses of rigid-body motions.

3.1.4 The Crystallographic Groups

The *crystallographic group* is a discrete subgroup of the *Euclidean group* $\mathrm{E}(n) := \mathrm{O}(n) \ltimes \mathbb{R}^n$, where $\mathrm{O}(n)$ is the *orthogonal group* consisting of all orthogonal real-valued $n \times n$ matrices (defined as in Eq. (3.13) for $n = 3$, but also allowing $\det R = -1$ there). If $n = 3$, a crystallographic group is commonly called a *space group*, and for $n = 2$ it is referred to as a *wallpaper group*.

3.1.4.1 Definitions and properties

Elements of a crystallographic group Γ can be expressed as pairs

$$\gamma \;=\; (R_\gamma, \boldsymbol{t}_\gamma + \boldsymbol{v}(R_\gamma)), \tag{3.41}$$

where $R_\gamma \in \mathbb{P}$ (a discrete *point group*, i.e. a subgroup of $\mathrm{O}(n)$), $\boldsymbol{t}_\gamma \in \mathbb{L}$ (a lattice in $\mathbb{R}^n$), and $\boldsymbol{v} : \mathbb{P} \to \mathbb{R}^n$. In particular, $\boldsymbol{v}$ is the translational part of a screw-displacement lattice motion. In general, $\boldsymbol{v}$ satisfies the *co-cycle identities*

- $\boldsymbol{v}(\mathbb{I}) = \boldsymbol{0}$; and
- $\boldsymbol{v}(R_{\gamma_1} R_{\gamma_2}) \;=\; (R_{\gamma_1} \boldsymbol{v}(R_{\gamma_2}) + \boldsymbol{v}(R_{\gamma_1})) \bmod T$, where $T := \{\mathbb{I}\} \ltimes \mathbb{L} \lhd \Gamma$ is the normal subgroup of pure translations in Γ.

The "mod T" removes components in the sum that are in T, which is analogous to $(3 + 8) \bmod 5 = 1$ in modulo-5 arithmetic.

If an element $\gamma \in \Gamma - \{e\}$ is of *finite order* (i.e. if there exists an $m \in \mathbb{N}$ such that $\gamma^m = e$), it is called a *torsion element*. The group Γ is called *torsion-free* (or a *Bieberbach group*) if it is free of torsion elements. This is equivalent to the property that no element $\gamma \in \Gamma$ other than the identity e has a *fixed point* (i.e. a

point $\boldsymbol{p} \in \mathbb{R}^n$ with $\gamma\boldsymbol{p} = \boldsymbol{p}$). If $\boldsymbol{v} \equiv \boldsymbol{0}$ in Eq. (3.41), then Γ can be written as the semi-direct product $\Gamma = \mathbb{P} \ltimes \mathbb{L}$ and the group is called *symmorphic*. Of the 230 possible types of space groups, 73 can be decomposed in this way, which are called the *symmorphic space groups*.

For a symmorphic space group Γ, a Bieberbach subgroup $\Gamma_B < \Gamma$ with minimum index in Γ is the subgroup T of primitive translations; for many non-symmorphic space groups, on the other hand, there is a Bieberbach subgroup Γ_B with index $[\Gamma : \Gamma_B] < [\Gamma : T]$ allowing for a decomposition of Γ as a group product [32]

$$\Gamma = \Gamma_B S := \{\gamma_B s \,|\, \gamma_B \in \Gamma_B, s \in S\}, \tag{3.42}$$

where $S < \Gamma$ is a proper subgroup of $\mathbb{P} \ltimes \{\boldsymbol{0}\} < \mathrm{E}(3)$, and thus $\Gamma_B \cap S = \{e\}$.

3.1.4.2 Concrete examples for 2D wallpaper groups and coset/double-coset decomposition

As introduced above, the 2D crystallographic groups are also referred to as wallpaper groups, since the resulting transformations of a single feature is analogous to wallpaper patterns. Throughout this dissertation, five typical wallpaper symmetry groups, i.e. p_1, p_2, p_3, p_4 and p_6 are of the most interests. The group elements are summarized as follows.

- p_1: consists solely of translations, constituting a parallelogrammatic lattice (the angle between axes is arbitrary) in the x-y plane;
- p_2: consists of planar translations as in p_1, and also contains a rotation of order two (i.e. with angle π);

- p_3: has a hexagonal translation lattice, and contains rotations of order three (rotation angles $\frac{2}{3}\pi$ and $\frac{4}{3}\pi$);

- p_4: contains rotations of order four (i.e. with angles $\frac{\pi}{2}$, π, and $\frac{3}{2}\pi$), as well as translations in a square lattice;

- p_6: has a hexagonal translation lattice, and contains rotations of order six (rotation angles $\frac{\pi}{3}$, $\frac{2}{3}\pi$, π, $\frac{4}{3}\pi$, and $\frac{5}{3}\pi$).

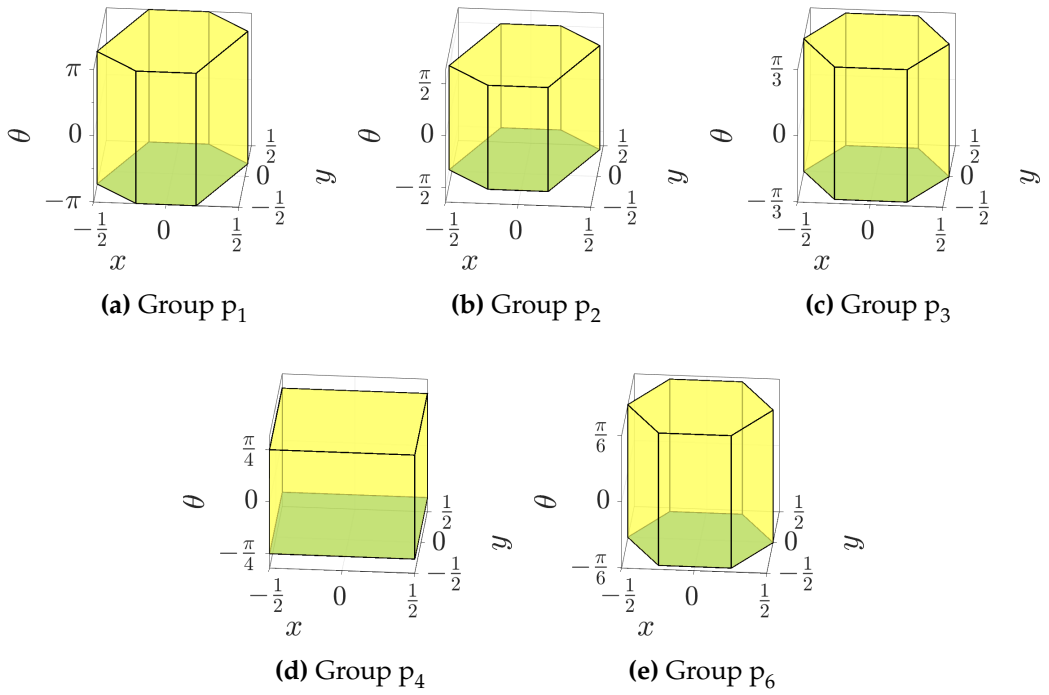
The cosets of $\mathrm{SE}(2)$ with respect to these subgroups can be computed and their corresponding fundamental domains $F_{\mathrm{p}_i\backslash\mathrm{SE}(2)}$ $(i = 1, 2, 3, 4, 6)$ are constructed using Eq. (3.11), as shown in Fig. 3.4 [61], [161]. The plots are in exponential coordinates, with the x and y axes representing translations along x and y directions in $\mathbb{R}^2$ and the z axis representing the rotation angle $\theta \in (-\pi, \pi)$. Euclidean distance on $\mathbb{R}^3$ is used here as the metric for computations. Note that this metric is left- but not right-invariant (there is no bi-invariant metric on $\mathrm{SE}(2)$). Therefore, the fundamental domains shown in Fig. 3.4 are actually Voronoi rather than Voronoi-like cells.

Below is a summary of geometric features of fundamental domains $F_{\mathrm{p}_i\backslash\mathrm{SE}(2)}$, where $i = 1, 2, 3, 4, 6$.

- $F_{\mathrm{p}_1\backslash\mathrm{SE}(2)}$: a box with (irregular) hexagonal shape in the x-y plane with height 2π;

- $F_{\mathrm{p}_2\backslash\mathrm{SE}(2)}$: also has a hexagonal shape in the translational plane, but the height is only π (from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$);

- $F_{\mathrm{p}_3\backslash\mathrm{SE}(2)}$: has a regular hexagonal shape in the translational plane, with a height of $\frac{2}{3}\pi$ (from $-\frac{\pi}{3}$ to $\frac{\pi}{3}$);



(a) Group p_1 **(b)** Group p_2 **(c)** Group p_3

(d) Group p_4 **(e)** Group p_6

Figure 3.4: Center Voronoi cell $F_{p_i \backslash SE(2)}$ for certain instances of the wallpaper groups (figures redrawn from [158], [163]).

- $F_{p_4 \backslash SE(2)}$: has the shape of a square in the translational plane, its height being $\frac{\pi}{2}$ (from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$);

- $F_{p_6 \backslash SE(2)}$: has a regular hexagonal shape in the translational plane, with a height of $\frac{\pi}{3}$ (from $-\frac{\pi}{6}$ to $\frac{\pi}{6}$).

3.2 Fundamental Geometric Tools

This section introduces several useful concepts and algorithms in geometry. The closed-form Minkowski sum and difference between an ellipsoid and any convex differentiable surface in $\mathbb{R}^n$ are derived at first. Then, the computation process for *minimum volume concentric ellipsoid (MVCE)* is algebraically derived.

Finally, the concepts of the Kinematics of Containment (KC) is discussed.

3.2.1 Closed-form Minkowski Operations Between an Ellipsoid and a General Convex Differentiable Surface Embedded in $\mathbb{R}^n$

The Minkowski sum and difference of two point sets (or bodies) centered at the origin, A and B in $\mathbb{R}^n$, are defined respectively as [14]

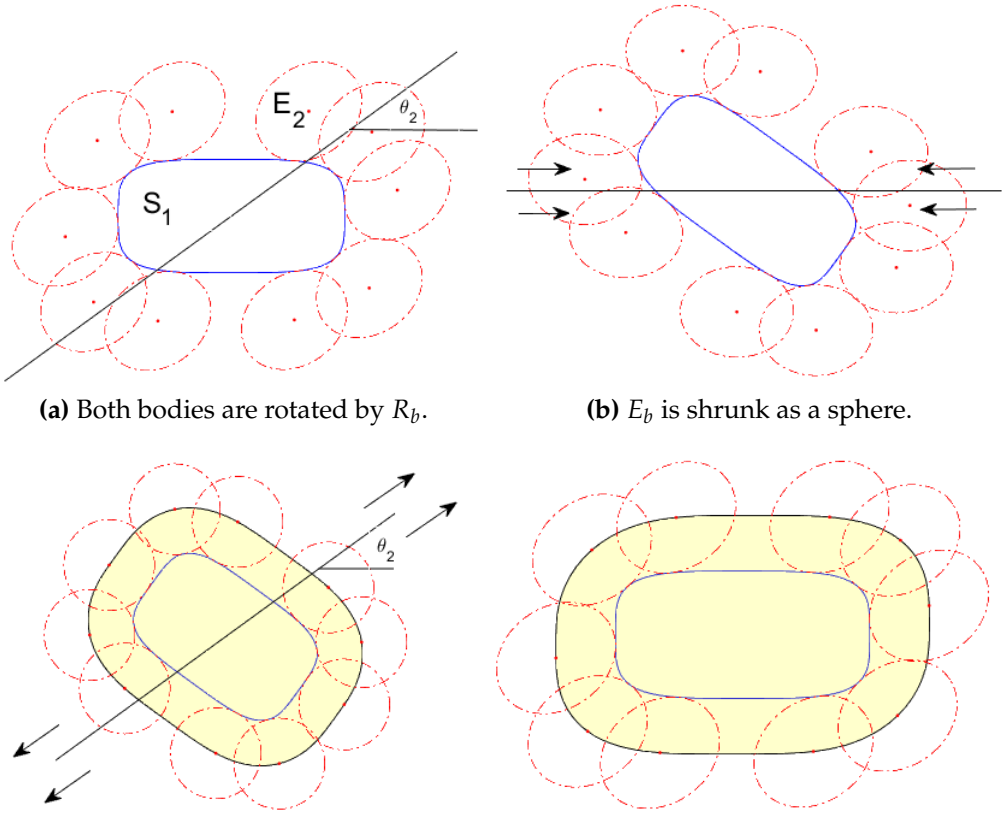
$$\begin{aligned} A \oplus B &\doteq \{a + b \,|\, a \in A, b \in B\}, \quad \text{and} \\ A \ominus B &\doteq \{p \,|\, p + B \subseteq A\}. \end{aligned} \tag{3.43}$$

Alternatively, the Minkowski difference of two bodies can be defined relative to the Minkowski sum as the body $A' = A \ominus B$ where $A = A' \oplus B$. While the definition is relatively simple, calculating useful representations of Minkowski operations can be difficult and computationally expensive, especially when the bodies are non-convex and the boundary of these regions requires an explicit representation. Also, using the fact that

$$\text{If } A = A_1 \cup A_2, \text{ then, } A \oplus B = (A_1 \oplus B) \cup (A_2 \oplus B), \tag{3.44}$$

convex decomposition is an effective solution to obtain the Minkwoski sums between two non-convex object [146].

It has been observed previously in [164] that the Minkowski sum and difference between two ellipsoids can be parameterized in closed-form. The computational procedure can be further extended when one ellipsoid is substituted by an arbitrary convex differentiable surface embedded in n-dimensional



(a) Both bodies are rotated by R_b.

(b) E_b is shrunk as a sphere.

(c) An offset surface in shrunk space.

(d) Transform back and obtain $S_a \oplus E_b$.

Figure 3.5: Algorithm for obtaining the characterizations of the Minkowski sum between a superquadric surface S_a and an ellipsoid E_b.

Euclidean space ($\mathbb{R}^n$) for which closed-form parametric and implicit descriptions are known. Figure 3.5 illustrates the computational process, and the algebraic derivations are shown as follows.

Assume that S_a is a convex and differentiable hyper-surface embedded in $\mathbb{R}^n$, with implicit and parametric forms being

$$\Phi(\boldsymbol{x}) = 1 \text{ and } \boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\psi}), \tag{3.45}$$

where $\Phi(\boldsymbol{x})$ is a real-valued differentiable function of $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{f}(\boldsymbol{\psi})$ is a differentiable n-dimensional vector-valued functions of $\boldsymbol{\psi} = [\psi_1, \psi_2, ..., \psi_{n-1}]^\top \in \mathbb{R}^{n-1}$. Let E_b be an ellipsoid in general orientation in $\mathbb{R}^n$, with semi-axis lengths $\boldsymbol{b} = [b_1, b_2, ..., b_n]^\top$. Then, the implicit and explicit equations are of the form

$$\boldsymbol{x}^\top B^{-2} \boldsymbol{x} = 1 \text{ and } \boldsymbol{x} = B\mathbf{u}(\boldsymbol{\psi}), \tag{3.46}$$

where $B = R_b \Lambda(\boldsymbol{b}) R_b^\top$ is the shape matrix of E_b where $R_b \in \mathrm{SO}(n)$ denotes the orientation of the ellipsoid, and $\Lambda(\boldsymbol{b})$ is a diagonal matrix with the semi-axis length b_i at the (i, i) entry. Here $\mathbf{u}(\boldsymbol{\psi})$ is the standard parameterization of the n-dimensional hyper-sphere.

By applying an affine transformation, i.e.

$$T = R_b \Lambda(r/\boldsymbol{b}) R_b^\top, \tag{3.47}$$

the ellipsoid can be shrunk into a sphere with radius $r = \min\{b_1, b_2, ..., b_n\}$, and the surface S_a can be transformed as $\boldsymbol{x}' \doteq T\boldsymbol{x}$. The affine transformation matrix is symmetric and positive definite since $\Lambda(r/\boldsymbol{b})$ is diagonal and positive definite.

The implicit expression for the "shrunk" S_a, denoted as S'_a, is $\Phi(T^{-1}\boldsymbol{x}') = 1$. Then the Minkowski sum between S'_a and E'_b (now is a sphere), is obtained by computing the boundary of the offset surface with offset radius r as

$$\boldsymbol{x}_{ofs} = \boldsymbol{x}' + r\mathbf{n}', \tag{3.48}$$

where $\mathbf{n}' = \frac{\nabla_{x'}\Phi(T^{-1}\boldsymbol{x}')}{\|\nabla_{x'}\Phi(T^{-1}\boldsymbol{x}')\|}$ is the outward normal of the surface. With the help

of the chain rule for derivatives, the numerator can be further simplified as

$$\nabla_{x'}\Phi(T^{-1}x')\Big|_{x'=Tx} = T^{-\top}\nabla_x\Phi(x). \tag{3.49}$$

In the special case here, T is a symmetric positive definite matrix, therefore, $T^{-\top} = (T^{-1})^{\top} = (T^{\top})^{-1} = T^{-1}$. Then the Minkowski sum between the original surface S_a and ellipsoid E_b can be given by "stretching" the transformed space back, using inverse affine transformation, as

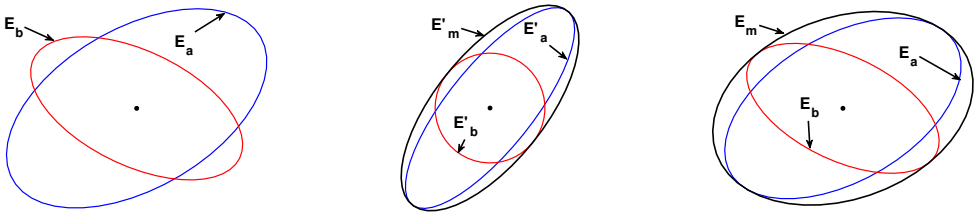
$$\begin{aligned} x_{mb} = T^{-1}x_{ofs} &= T^{-1}\left(Tx + r\frac{T^{-\top}\nabla_x\Phi(x)}{\|T^{-\top}\nabla_x\Phi(x)\|}\right) \\ &= x + r\frac{T^{-2}\nabla_x\Phi(x)}{\|T^{-1}\nabla_x\Phi(x)\|}. \end{aligned} \tag{3.50}$$

Further simplification for Eq. (3.50) shows that the radius can be eliminated. And using the fact that the magnitude of a vector is preserved under rotation, the general form of the closed-form Minkowski sums between an ellipsoid and a general convex differentiable surface in $\mathbb{R}^n$ can be computed as

$$x_{mb} = x + \frac{R_b\Lambda^2(b)R_b^{\top}\nabla_x\Phi(x)}{\|R_b\Lambda(b)R_b^{\top}\nabla_x\Phi(x)\|} := x + x_n \tag{3.51}$$

The Minkowski difference $S_a \ominus E_b$ therefore can be obtained by switching the plus signs in Eqs. (3.50) and (3.51) to minus. However, for the Minkowski difference, a "curvature constraint" should be satisfied: after the "shrinking" operation, the curvature of every point on the transformed surface S_a' should be smaller than the curvature of the transformed ellipsoid E_b'.

For specific applications, superquadric surfaces are chosen to encapsulate the objects in the environment. The explicit expressions for the Minkowski



(a) Two concentric ellipses, E_a and E_b.

(b) Shrink E_a into a circle E'_a, and bound with a new ellipse E'_m.

(c) Transform the whole space back to get MVCE E_m.

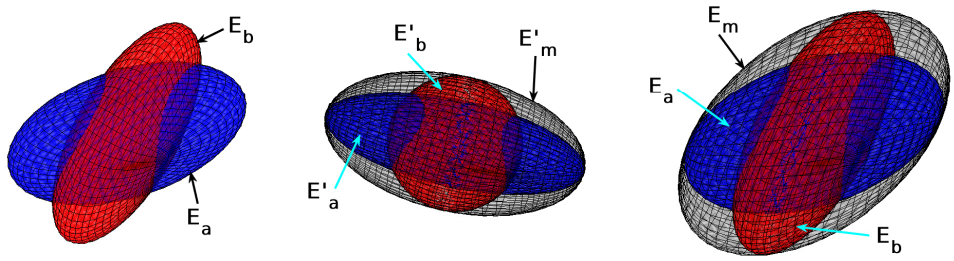
Figure 3.6: Computational procedure for minimum volume concentric ellipse that covers two ellipses in 2D.

operations between an ellipsoid and a superquadric surface in both 2D and 3D are derived in Appendix A.1.

3.2.2 The Minimum Volume Concentric Ellipsoid Enclosing Two n-dimensional Ellipsoids at the Same Center

When two ellipsoids are fixed at the same center, a larger concentric ellipsoid, denoted as the *minimum volume concentric ellipsoid (MVCE)* can be computed in closed form. The computational procedure is derived as follows as well as visualized in Fig. 3.6. Note that the idea here is inspired by [120], which provides the equivalent computations for a maximum volume concentric ellipsoid that is covered by two ellipsoids.

Suppose the two ellipsoids, E_a and E_b, have semi-axis lengths $\boldsymbol{a}$ and $\boldsymbol{b}$ respectively. The shape matrices can be expressed as $A = R_a\Lambda^{-2}(\boldsymbol{a})R_a^\top$ and $B = R_b\Lambda^{-2}(\boldsymbol{b})R_b^\top$, and the parametric expressions are $\boldsymbol{x}_a = R_a\Lambda(\boldsymbol{a})\mathbf{u}$ and $\boldsymbol{x}_b = R_b\Lambda(\boldsymbol{b})\mathbf{u}$ respectively, where R is the rotation matrix, $\Lambda(\cdot)$ is a diagonal matrix and $\mathbf{u}$ is the parameterized expression for a unit sphere.



(a) Two concentric ellipsoids, E_a and E_b.

(b) Shrink E_a into a sphere E'_a, and bound with a new ellipsoid E'_m.

(c) Transform the whole space back to get MVCE E_m.

Figure 3.7: Computational procedure for minimum volume concentric ellipsoid that covers two ellipsoids in 3D.

It has been shown, when deriving the closed-form Minkowski operations, that one ellipsoid (i.e. E_b) can be shrunk into a sphere (E'_b) via the affine transformation $T = R_b\Lambda(r/\boldsymbol{b})R_b^\top$, where r is the radius. Then shape matrix for E_a in shrunk space, i.e. E'_a, can be computed as

$$A' = T^{-1}R_a\Lambda^{-2}(\boldsymbol{a})R_a^\top T^{-1}, \tag{3.52}$$

and the singular value decomposition (SVD) returns its semi-axis lengths and orientation, i.e $\boldsymbol{a}'$ and R'_a respectively.

Now the problem becomes that of finding an ellipsoid E'_m that fully encloses an ellipsoid E'_a and a sphere E'_b with the same center. The semi-axes directions are aligned with these of E'_a and E'_b, and their lengths are set as the maximum values between elements in $\boldsymbol{a}'$ and the radius r of E'_b. Then the shape matrix for E'_m can be obtained as

$$M' = R'_a\Lambda^{-2}(\max(\boldsymbol{a}', r))R_a'^\top. \tag{3.53}$$

Finally, the space is stretched back by applying the inverse affine transformation T^{-1}, and get the shape matrix of the MVCE, E_m, as

$$M = TM'T = R_b\Lambda(r/\boldsymbol{b})R_b^\top R'_a\Lambda^{-2}(\max(\boldsymbol{a}', r))R_a'^\top R_b\Lambda(r/\boldsymbol{b})R_b^\top. \quad (3.54)$$

The semi-axis lengths and the orientation can also be obtained by SVD of M. All the above derivations are valid in any dimension, since no dimensional restrictions are applied in either the shape matrices or the affine transformation matrices. The volume minimality of the resulting concentric ellipsoid E_m is proved in Appendix A.3.

Furthermore, this computational procedure can be performed iteratively if there are multiple concentric ellipsoids. For example, the MVCE that enclose the previous two ellipsoids, along with the next ellipsoid, can be enclosed by a new MVCE. Therefore, the final resulting ellipsoid encapsulates all the original set of ellipsoids.

3.2.3 The Kinematics of Containment

The Kinematics of Containment (KC) addresses the range of allowable motions for one convex body in $\mathbb{R}^n$ being completely inside of another, and provides a simple expression to compute the volume of such motions in the group of rigid-body motions, $\mathrm{SE}(n)$ [35]. The derivations are based on the Principal Kinematics Formula (PKF) [138] from the field of integral geometry.

PKF studies the range of possible motions when two convex bodies, K_a and K_b in $\mathbb{R}^n$, intersect to each other. Such a range of motions can be characterized by the indicator function $\iota(\cdot)$, which is defined as 1 when the argument is

nonempty and 0 otherwise. Therefore, when the two convex bodies intersect, i.e. $\iota(g \cdot K_a \cap K_b) = 1$ (assuming K_a is moving and K_b is fixed, $g \in \mathrm{SE}(n)$ and $g \cdot K_a \doteq RK_a + \boldsymbol{t}$ defines the group action on the rigid body K_a, where $R \in \mathrm{SO}(n)$ denotes the orientation and $\boldsymbol{t} \in \mathbb{R}^n$ is the translation of the body), the volume of such motions can be computed as, when $n = 2$,

$$\begin{aligned} V_{PKF}^{\mathrm{SE}(2)} &= \int_{\mathrm{SE}(2)} \iota(g \cdot K_a \cap K_b) dg \\ &= 2\pi[\mathcal{A}(K_a) + \mathcal{A}(K_b)] + \mathcal{P}(\partial K_a) \cdot \mathcal{P}(\partial K_b), \end{aligned} \tag{3.55}$$

where $\mathcal{A}$ and $\mathcal{P}$ are the area and perimeter of a planar object respectively, and dg is the bi-invariant integral measure for $\mathrm{SE}(n)$. When $n = 3$,

$$\begin{aligned} V_{PKF}^{\mathrm{SE}(3)} &= \int_{\mathrm{SE}(3)} \iota(g \cdot K_a \cap K_b) dg \\ &= 8\pi^2[\mathcal{V}(K_b) + \mathcal{V}(K_a)] + 2\pi\mathcal{F}(\partial K_b)\mathcal{M}(\partial K_a) \\ &\quad + 2\pi\mathcal{F}(\partial K_a)\mathcal{M}(\partial K_b), \end{aligned} \tag{3.56}$$

where $\mathcal{V}$ is the volume of a spatial body, and $\mathcal{F}$ and $\mathcal{M}$ are the surface area and the integral of mean curvature of the bounding surface enclosing a spatial body respectively.

Different from PKF, the KC theory deals with the case when the moving K_a is completely contained inside K_b, i.e. $\iota(g \cdot K_a \subseteq K_b) = 1$. Then the volume of all possible motions of K_a can be expressed as, when $n = 2$,

$$\begin{aligned} V_{KC}^{\mathrm{SE}(2)} &= \int_{\mathrm{SE}(2)} \iota(g \cdot K_a \subseteq K_b) dg \\ &= 2\pi[\mathcal{A}(K_a) + \mathcal{A}(K_b)] - \mathcal{P}(\partial K_a) \cdot \mathcal{P}(\partial K_b), \end{aligned} \tag{3.57}$$

and when $n = 3$,

$$\begin{aligned} V_{KC}^{\mathrm{SE}(3)} &= \int_{\mathrm{SE}(3)} \iota(g \cdot K_a \subseteq K_b) dg \\ &= 8\pi^2[\mathcal{V}(K_b) - \mathcal{V}(K_a)] - 2\pi\mathcal{F}(\partial K_b)\mathcal{M}(\partial K_a) \\ &\quad + 2\pi\mathcal{F}(\partial K_a)\mathcal{M}(\partial K_b). \end{aligned} \tag{3.58}$$

The general KC theory provides the above clean and simple expressions for volume computations of the allowable motion space for arbitrary convex bodies.

Moreover, the volume expressions for the PKF and KC are also related to the Minkowski sum and difference between the two convex bodies with fixed orientations. Concretely and taking the volume of KC C-space as an example, the volume expression can be rewritten, for $\mathrm{SE}(n)$, as [35]

$$\begin{aligned} V_{KC}^{\mathrm{SE}(n)} &= \int_{\mathrm{SE}(n)} \iota(g \cdot K_a \subseteq K_b) dg \\ &= \int_{\mathrm{SO}(n)} \int_{\mathbb{R}^n} \iota(\boldsymbol{t} \cdot (R \cdot K_a) \subseteq K_b) d\boldsymbol{t} dR \\ &= \int_{\mathrm{SO}(n)} \mathcal{V}(K_b \ominus (R \cdot K_a)) dR, \end{aligned} \tag{3.59}$$

where $\ominus$ denotes the Minkowski difference between two bodies.

The integrand is the volume of the Minkowski difference [7] between K_b and a rotated version of K_a, and the total volume of KC C-space can be computed by integrating over the rigid-body rotation group. Throughout this article, Eq. (3.59) is used as a reference to evaluate the relative volumes of different lower bounds, denoted as the volume of the "actual" KC C-space.

Note that Eq. (3.57)-(3.59) that extend from PKF to KC are heuristic with the following conditions being hold: the moving K_a is required to kiss the inner wall of K_b at one point while being fully contained at all orientations; in addition, the Minkowski difference between the two convex bodies is also required to exist at every orientation and be convex.

Part II

Group-Theoretic Approaches to Robotic Motion Quantization and Calibration

Chapter 4

Quantizing Euclidean Motions via Double-Coset Decomposition

This chapter introduces the concept of a *motion alphabet* to express the motion primitives of robots, with which a dictionary of physical actions are built. Since an alphabet forms a discrete representation for a continuous sequence of actions, this dictionary is also called a *quantization* of robot motions. The construction of such a dictionary is based on two major aspects of Lie groups: (1) Decomposition of the group into cosets, double cosets and the corresponding fundamental domains; and (2) Construction of such fundamental domains as Voronoi or Voronoi-like cells. Another important goal of this chapter is to develop efficient algorithms to round off continuous motions to their nearby discrete representatives, which solves for the *signal-to-symbol* problems. With the above two goals, actions of smart robots in the world can be approximated by finite sequences of characters, thereby forming the foundation of a language in which to articulate robot motions.

4.1 Introduction

The classical *signal-to-symbol* problem in artificial intelligence (AI) seeks to interpret the continuous world into countable classes of discrete symbols, such as the letters in an alphabet. This is analogous as the inverse problem of what the genetic code does, where the finite set of alphabets $\{A, C, G, T\}$ encodes the morphology and metabolism of every living creature in the continuous world [41]. Discrete alphabets form the basis for all human language [141], which depends on how much information can be conveyed with a given number of symbols, and how difficult it is to express the symbols.

In the field of robotics, a typical example using this discrete interpretation for continuous actions is robot motion planning, which is one of the fundamental topics addressed for decades. In general, motion planning seeks to answer the query: "how to plan a path that guides the robot from a given start pose to an end pose subject to some geometric or dynamical constraints". A popular way that has been applied for decades is to build a *roadmap* [84], which basically is a graph structure consisting of discrete valid vertices and edges. Once a query is submitted, graph searching algorithms give a valid optimal sequence of motions from the roadmap. Although a roadmap method is able to answer multiple searching queries, when the environment is changing, those vertices or edges information need to be updated. To some extent, quantizing a continuous motion of the robot from a pre-defined motion alphabet is closely related to the roadmap concept. But the advantage lies on the storage and representation of the motion sequences. All other elements of our motion alphabet will be written as a product of the form $\gamma\delta$ where $\gamma \in \Gamma$, a

crystallographic space group, and $\delta \in \Delta$, a finite group of rotational symmetries. Therefore, once a rich library of alphabets is built a priori, representing motions in different environment subject to varies constraints is just a matter of combinations and ordering of the alphabet indices.

The combination of letters in the motion alphabet raises the basic problem that is addressed in this chapter: approximate (or round off) continuous objects and actions within a discrete descriptive framework. A specific kind of round-off (of Euclidean motions) is proposed and verified, which makes it precise in the context of motions of objects and intelligent agents in the world.

The major contributions are listed as follows:

- A signal-to-symbol framework for Euclidean motions based on double-coset decomposition is proposed and its properties are analyzed.
- Concrete formulations of the motion alphabet are constructed through crystallographic symmetry.
- Decoding algorithms via coarse-to-fine double-coset decompositions are proposed and numerically verified.
- A hybrid search method that incorporates some existing sampling methods for rotation group with good dispersion or discrepancy properties has been proposed and verified.

The remainder of this chapter is structured as follows. Section 4.2 introduces the concept of a motion alphabet by discretizing the group of rigid-body motions (i.e. $\mathrm{SE}(3)$) via a fine double-coset decomposition based on a crystallographic space group and the subgroup of rotational symmetries. Section

4.3 presents solutions to the decoding problem by introducing a coarse-to-fine searching algorithm. In Section 4.4, the proposed quantization algorithms are compared with existing methods for sampling rotations and Euclidean motions. It is shown how the motion alphabet can be used as a geometric data structure to enhance the speed of these other motion-approximation methods.

4.2 Alphabets of Euclidean Motions Based on the Double-Coset Decomposition

The Euclidean motions of a robot in the physical world can be described as a sequence of elements in $\mathrm{SE}(3)$. Such continuous paths can be viewed as *signals*. Some typical *signal-to-symbol* decoding solutions convert the observations in the continuous world to coarsified representations, i.e. *symbols*. Examples of these transformations include the language we are speaking everyday [70] and cognitive signal processing systems [155] and classical AI approaches [128]. A natural way to quantize the space of motions is to first discretize time, thereby reducing the dimensions to a finite number. Then, each sampled pose on the path is replaced by a rounded-off version from a discrete set. An essential requirement for such a set of representative symbols is its uniformity: the higher uniformity of the set, the smaller round-off errors can be achieved. In the case of $\mathrm{SE}(3)$, it is hard to obtain a uniform discrete subset, mainly due to a nontrivial uniformly $\mathrm{SO}(3)$ sampling. In general, a uniform sampling on the parameter space might not result in a uniform sampling of $\mathrm{SO}(3)$, because of the metric distortions introduced in the nonlinear mapping [163], [167]. Thus, an novel idea of double-coset decomposition for $\mathrm{SE}(3)$ is

proposed as a solution for the discretization. In the following, several choices of decomposition are presented and analyzed.

4.2.1 Double-coset decomposition by a finite space group and rotation group

Building on the ideas introduced in the context of protein-packing models in X-ray crystallography [26], [27], [32], [33], [161], a fine space group Γ can be augmented with a discrete rotation group Δ. In particular, one of the finest 3D space groups is $\Gamma = \mathrm{P432}$ which has a total of 24 rotational elements corresponding to the rotational symmetry operations of a cube [61]. Furthermore, let Δ denote the group of rotational symmetries of the icosahedron, which has 60 elements. Since $\Gamma, \Delta < \mathrm{SE}(3)$ and $\Gamma \cap \Delta = \{e\}$, the double-coset space $\Gamma\backslash\mathrm{SE}(3)/\Delta$ is a compact Riemannian manifold. It is then possible to define a compact fundamental domain $F_{\Gamma\backslash\mathrm{SE}(3)/\Delta} \subset \mathrm{SE}(3)$ using Eq. (3.12). Restricting the quotient map from $\mathrm{SE}(3)$ to $\Gamma\backslash\mathrm{SE}(3)/\Delta$ to the fundamental domain $F_{\Gamma\backslash\mathrm{SE}(3)/\Delta}$ then gives a one-to-one mapping from $F_{\Gamma\backslash\mathrm{SE}(3)/\Delta}$ to $\Gamma\backslash\mathrm{SE}(3)/\Delta$. Moreover, the action of Γ (from the left) and Δ (from the right) gives a way to tile $\mathrm{SE}(3)$ with disjoint shifts of $F_{\Gamma\backslash\mathrm{SE}(3)/\Delta}$, because

$$\mathrm{SE}(3) = \bigcup_{\gamma\in\Gamma} \bigcup_{\delta\in\Delta} \gamma F_{\Gamma\backslash\mathrm{SE}(3)/\Delta}\delta. \tag{4.1}$$

When the fundamental domain $F_{\Gamma\backslash\mathrm{SE}(3)/\Delta}$ is constructed using Eq. (3.12), the identity element e locates at its center. Therefore, the tiling in Eq. (4.1) has the effect of sampling each center point by moving from e to $\gamma\delta$ where $\gamma \in \Gamma$ and $\delta \in \Delta$. In other words, the product $\Gamma \times \Delta < \mathrm{SE}(3) \times \mathrm{SE}(3)$ can be used as

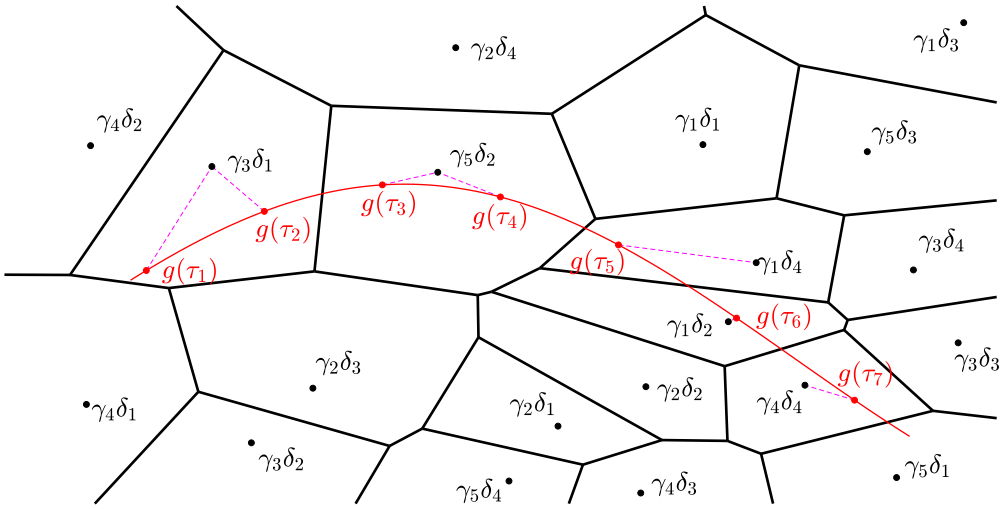


Figure 4.1: Discretizing a continuous motion trajectory g at times $\tau_1, \ldots, \tau_7$ using the alphabet $\Gamma \times \Delta$ (conceptual plot). After discretization the continuous motion can be expressed as the sentence (γ_3, δ_1), (γ_3, δ_1), (γ_5, δ_2), (γ_5, δ_2), (γ_1, δ_4), (γ_1, δ_2), (γ_4, δ_4) (figure redrawn from [158]).

a quantized version of $\mathrm{SE}(3)$. An important advantage of this quantization framework, i.e. Eq.(4.1), is that the shifted fundamental domains $\gamma F_{\Gamma\backslash \mathrm{SE}(3)/\Delta}\delta$ will all have the same volume, i.e.

$$\mu(\gamma F_{\Gamma\backslash \mathrm{SE}(3)/\Delta}\delta) \;=\; \mu(F_{\Gamma\backslash \mathrm{SE}(3)/\Delta}),\; \forall\,(\gamma,\delta) \in \Gamma \times \Delta, \tag{4.2}$$

where μ is the (left- and right-invariant) Haar measure on the (unimodular) Lie group $\mathrm{SE}(3)$.

The alphabet defined by $\Gamma \times \Delta$ is infinite, but by limiting the extent of translations to be contained in a bounded region, it becomes finite. This means that continuous trajectories can be translated into a finite string of alphabet characters (Fig. 4.1). This opens up the possibility to map these quantified trajectories into words expressed in a natural language.

4.2.2 More examples of decomposition

The decomposition of $\mathrm{SE}(3)$ using the proposed method relies on the choice of discrete subgroups $\Gamma, \Delta < \mathrm{SE}(3)$. When the chosen subgroups satisfy $\Gamma \cap \Delta = \{e\}$, a fundamental domain can be constructed as in Eq. (3.12), which decomposes $\mathrm{SE}(3)$ as in Eq. (4.1). There are many ways to choose $F_{\Gamma\backslash\mathrm{SE}(3)}$ and $F_{\Gamma\backslash\mathrm{SE}(3)/\Delta}$, as explained in [33]. A particularly simple choice is the Cartesian product

$$F_{\Gamma\backslash\mathrm{SE}(3)/\Delta} := F_{\mathbb{P}\backslash\mathrm{SO}(3)/\Delta} \times F_{\mathbb{L}\backslash\mathbb{R}^3}, \tag{4.3}$$

where $\mathbb{P} \cong \frac{\Gamma}{T}$ is the point group of Γ and $\mathbb{L}$ the lattice of primitive translations. Alternatively, given a decomposition in Eq. (3.42), another natural choice is

$$F_{\Gamma\backslash\mathrm{SE}(3)/\Delta} := F_{S\backslash\mathrm{SO}(3)/\Delta} \times F_{\Gamma_B\backslash\mathbb{R}^3}. \tag{4.4}$$

Here $S\backslash\mathrm{SO}(3)/\Delta$ and $\Gamma_B\backslash\mathbb{R}^3$ are not coset or double-coset spaces, because $S \not< \mathrm{SO}(3)$ and $\Gamma_\mathrm{B} \not< \mathbb{R}^3$. However, they are *orbit spaces* consisting respectively of *orbits* $SR\Delta := \{sR\delta : s \in S,\ \delta \in \Delta\}$ and $\Gamma_B\boldsymbol{x} := \{\gamma_\mathrm{B}\boldsymbol{x} : \gamma_\mathrm{B} \in \Gamma_\mathrm{B}\}$ ($R \in \mathrm{SO}(3)$, $\boldsymbol{x} \in \mathbb{R}^3$). The fundamental domains $F_{S\backslash\mathrm{SO}(3)/\Delta}$ and $F_{\Gamma_B\backslash\mathbb{R}^3}$ above can be constructed by choosing exactly one point per orbit.

4.3 The Coarse-to-Fine Decoding Algorithms

Once a motion alphabet introduced in the previous section is built, the next important problem is to decode the sampled pose into its nearest symbol. This section proposes a *coarse-to-fine* decoding algorithm to round off a specific pose into its closest representative. Two typical and important cases are specifically

studied.

4.3.1 The Case of Planar-Motion Alphabets Based on the Wallpaper Groups

As an example of a planar-motion alphabet similar to the one described in Section 4.2, consider the double-coset space $\Gamma\backslash \mathrm{SE}(2)/\Delta$ with $\Gamma = \mathrm{p}_4$ and $\Delta = \mathrm{C}_{2n-1} \ltimes \{\mathbf{0}\} < \mathrm{SE}(2)$, where

$$\mathrm{C}_{2n-1} := \left\{ \begin{bmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{bmatrix} \,\middle|\, \theta_j = \frac{2\pi j}{2n-1},\ j = 0, \ldots, 2(n-1) \right\} < \mathrm{SO}(2) \tag{4.5}$$

is the group of rotations of order $2n-1$ ($n \in \mathbb{N}$). Since in this choice, $\Gamma \cap \Delta = \{e\}$, the fundamental domain $F_{\Gamma\backslash \mathrm{SE}(2)/\Delta}$ is a Voronoi-like cell using Eq. (3.12). Figure 4.2 shows an visualization of such a fundamental domain when $n = 3$. In fact, because the left-invariant metric used here is also invariant under purely rotational actions from the right, the fundamental domain $F_{\Gamma\backslash \mathrm{SE}(2)/\Delta}$ is a classical Voronoi cell. Analogously as in Sec. 4.2, alphabet $\Gamma \times \Delta < \mathrm{SE}(2)^2$ can be used for an equi-volumetric quantization of $\mathrm{SE}(2)$. Because the scaling of the translational lattice in the wallpaper groups is arbitrary, the alphabet $\Gamma \times \Delta$ can be made arbitrarily fine by reducing the translational scaling in p_4 and increasing the parameter n above.

To illustrate the usage of the planar-motion alphabets constructed above, consider the $\mathrm{SE}(2)$ trajectory $g(\tau) := (R(\tau), \boldsymbol{t}(\tau))$, $\tau \in [0, 2\pi)$, where $R(\tau)$ is a rotation by an angle of τ and

$$\boldsymbol{t}(\tau) := \left[4\cos\tau,\ 6\Big(\frac{\tau}{2\pi}\Big) - 3 \right]^\top.$$

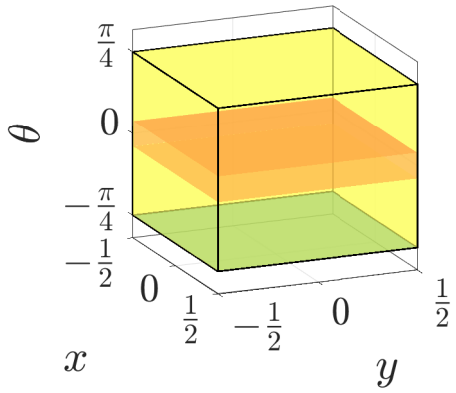


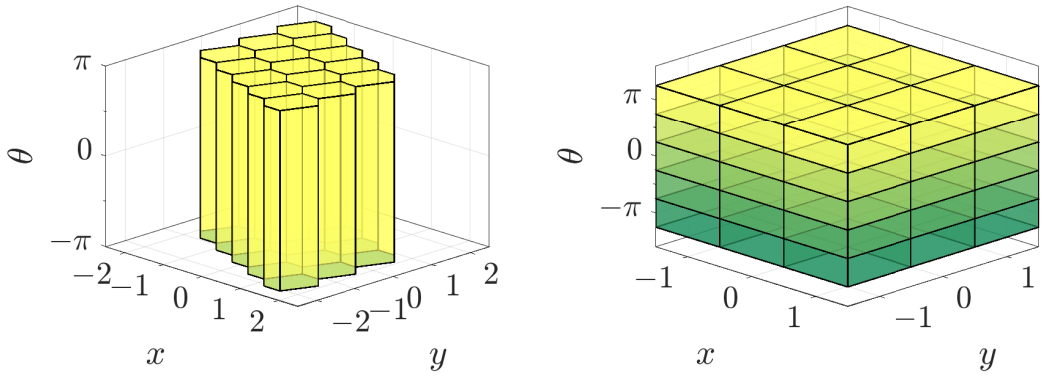
Figure 4.2: Center Voronoi cell $F_{\Gamma\backslash \mathrm{SE}(2)}$ in single-coset space (yellow outer region) based on an instance of the wallpaper group $\Gamma = \mathrm{p}_4$, and center Voronoi cell $F_{\Gamma\backslash \mathrm{SE}(2)/\Delta}$ in double-coset space (red inner region) with $\Delta = \mathrm{C}_5 \ltimes \{\mathbf{0}\}$ (figure redrawn from [158]).

g can be discretized at the five equidistant time points $\pi(1 + 2k/5)$, $k = -2, \ldots, 2$. As a motion alphabet for $\mathrm{SE}(2)$, $\Gamma \times \Delta = \mathrm{p}_4 \times \mathrm{C}_5$ is used. The elements of C_5 are denoted by δ_j with the index j as in Eq. (4.5). Let the elements of p_4 be denoted as $\gamma_{l,m,n}$, where $m \in \mathbb{Z}$ and $n \in \mathbb{Z}$ denote the translations in x and y direction respectively, and $l \in \{0, 1, 2, 3\}$ indicates a rotation by an angle of $l\pi/2$. The continuous motion trajectory g can now be expressed as the sentence $(\gamma_{2,3,-2}, \delta_3)$, $(\gamma_{2,-1,-1}, \delta_4)$, $(\gamma_{2,-4,0}, \delta_0)$, $(\gamma_{2,-1,1}, \delta_1)$, $(\gamma_{2,3,2}, \delta_2)$.

The decoding problem can be solved by a coarse-to-fine algorithm:

- For a given element $g \in \mathrm{SE}(2)$, find $\gamma \in \mathrm{p}_4$ such that $g \in \gamma F_{\Gamma\backslash \mathrm{SE}(2)}$.
- Search for the shifted fundamental domain $\gamma' F_{\Gamma\backslash \mathrm{SE}(2)/\Delta}\delta$ containing the pulled-back element $\gamma^{-1}g$ by a purely rotational search.

The first step is particularly easy in the case of p_4 when compared with,



(a) Shifted copies of $F_{\mathrm{p}_1\backslash\mathrm{SE}(2)}$ **(b)** Shifted copies of $F_{\mathrm{p}_4\backslash\mathrm{SE}(2)}$

Figure 4.3: Decompositions of $\mathrm{SE}(2)$ by p_1 and p_4, illustrated in exponential coordinates (conceptual plot), respectively (figure redrawn from [158]).

e.g. the group p_1 (with anisotropic translational lattice). In fact, as implied by Fig. 4.3, in the case of p_4 the above step can be realized by appropriately rounding off the translational components of g, as well as the rotation angle. In the case of p_1, on the other hand, distances of g to the Voronoi centers can be computed. And for the second step, the decomposition of g then reads $(\gamma\gamma')Q\delta$ with $Q = (\gamma\gamma')^{-1}g\delta^{-1} \in F_{\Gamma\backslash\mathrm{SE}(2)/\Delta}$. It is also possible to first treat the translational part of g, and then the rotational part.

4.3.2 The Case of Spatial-Motion Alphabets Based on the Pose Change Group

The choice for $F_{\Gamma\backslash\mathrm{SE}(3)/\Delta}$ in Eq. (4.3) allows us to bootstrap from of the fundamental domains for double-coset spaces of $\mathrm{SO}(3)$ discussed earlier. A trajectory of 3D Euclidean motions can be further described as a sequence of rotation and translation pairs, i.e. $g(\tau) = (R(\tau), \boldsymbol{t}(\tau))$ in $\mathrm{SO}(3) \times \mathbb{R}^3$ (a direct product rather than a semi-direct product). This is not merely to make things

easier – viewing pose change trajectories in this way has some advantages, as described in [31], where the direct product $\mathrm{SO}(3) \times \mathbb{R}^3$ is called the *pose change group*, and is denoted as $\mathrm{PCG}(3)$ [1]. By using $\mathrm{PCG}(3)$ rather than $\mathrm{SE}(3)$, the rotation and translation components of an Euclidean motion can be decoupled and considered separately. The following contexts first define and show some interesting properties of $\mathrm{PCG}(3)$, then dig into details about the decoding algorithm for the pure rotation case.

4.3.2.1 Decoupling rotation and translation parts using direct-product properties

The traditional rigid-body motion group $\mathrm{SE}(3)$ yields a semi-direct product, meaning that after the group product operation, the resulting translation component is coupled with the translation part, i.e. $(R_1, \boldsymbol{t}_1) \circ (R_2, \boldsymbol{t}_2) = (R_1 R_2, R_1 \boldsymbol{t}_2 + \boldsymbol{t}_1)$, where $R_1, R_2 \in \mathrm{SO}(3)$ and $\boldsymbol{t}_1, \boldsymbol{t}_2 \in \mathbb{R}^3$. This characterization of transformations has been treated as standards in robotic kinematics for decades. However, when describing the changes of poses as viewed in a third observer, it is not convenient in the sense that a conjugated motion has to be involved. Specifically, a three-indexed quantity is defined for a rigid-body motion as ${}^{O}_{A}H_{B}$, meaning that it moves from frame A to frame B as viewed from a third frame O, the conversion from the motions as viewed in frame A and B is a conjugation, i.e

$$\boxed{{}^{O}_{A}H_{B} = {}^{O}_{O}H_{A}\,{}^{A}_{A}H_{B}\,{}^{O}_{O}H^{-1}_{A}}. \tag{4.6}$$

[1] The code associated is available in https://github.com/ruansp/pcg_jmr2018.git

A more natural way to view this problem is to present a direct-product operation, and this is where PCG(3) fits in. If ${}^{O}_{A}R_{B} = {}^{O}_{O}R_{A}\,{}^{A}_{A}R_{B}\,{}^{O}_{O}R^{\top}_{A} \in \mathrm{SO}(3)$ and ${}^{O}_{A}\boldsymbol{t}_{B} \in \mathbb{R}^3$ are defined as the rotation and translation from frame A to B as view in O respectively, the combination of

$$\boxed{{}^{O}_{A}R_{C} = {}^{O}_{B}R_{C}\,{}^{O}_{A}R_{B} \text{ and } {}^{O}_{A}\boldsymbol{t}_{C} = {}^{O}_{B}\boldsymbol{t}_{C} + {}^{O}_{A}\boldsymbol{t}_{B}} \tag{4.7}$$

defines a direct product of the rotation and translation groups, i.e

$$\left({}^{O}_{A}R_{C}, {}^{O}_{A}\boldsymbol{t}_{C}\right) = \left({}^{O}_{B}R_{C}, {}^{O}_{B}\boldsymbol{t}_{C}\right) \cdot \left({}^{O}_{A}R_{B}, {}^{O}_{A}\boldsymbol{t}_{B}\right). \tag{4.8}$$

This direct-product defines the *pose change group*

$$\mathrm{PCG}(3) := \mathrm{SO}(3) \times \mathbb{R}^3. \tag{4.9}$$

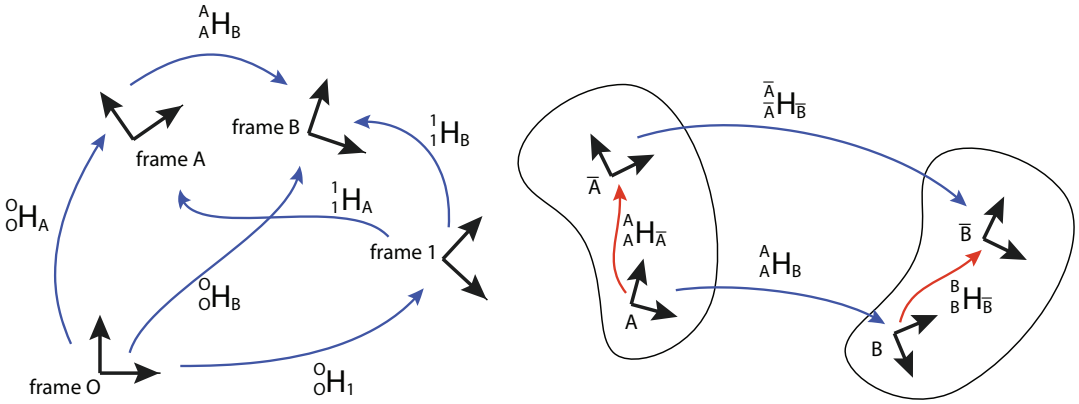
Using this direct-product operation, one can easily make changes of the observer's frame via the group action on the pose space. In particular, an action of PCG(3) on pose space can be defined as

$$(Q, \boldsymbol{\xi}) \odot (R, \mathbf{t}) \doteq \left(QRQ^{\top}, Q\mathbf{t}\right), \tag{4.10}$$

where $(Q, \_) \in \mathrm{PCG}(3)$. Note that the right hand side of (4.10) does not depend on $\boldsymbol{\xi} \in \mathbb{R}^3$. The action $\odot$ is associated with a change of observer frame from O to 1 as

$$\boxed{\left({}^{1}_{A}R_{B}, {}^{1}_{A}\mathbf{t}_{B}\right) = \left({}^{1}_{1}R_{0}, \mathbf{0}\right) \odot \left({}^{O}_{A}R_{B}, {}^{O}_{A}\mathbf{t}_{B}\right)}. \tag{4.11}$$

Figure 4.4a shows a conceptual plot for the transformations of the observer frame from O to 1.



(a) Transformation of the observer frame from O to 1.

(b) Conjugation resulting from changing body-fixed frames.

Figure 4.4: The view changes of observer and body-fixed frames.

In addition, a pose change can also be thought as modeling the motion of a frame that is attached to a rigid body. It can be interpreted that frame A is the body-fixed frame before motion of the body and frame B is where it goes to after the motion. Then, when changing the body-fixed frame which represents the rigid body from A to $\overline{A}$ before the motion and from B to $\overline{B}$ thereafter, the corresponding pose between the two new body frames, i.e. from $\overline{A}$ to $\overline{B}$, can be computed as

$$\boxed{\left({}^{O}_{\overline{A}}R_{\overline{B}}, {}^{O}_{\overline{A}}\mathbf{t}_{\overline{B}}\right) = \left(\mathbb{I}, {}^{O}_{B}\mathbf{t}_{\overline{B}}\right) \bullet \left({}^{O}_{A}R_{B}, {}^{O}_{A}\mathbf{t}_{B}\right) \bullet \left(\mathbb{I}, {}^{O}_{A}\mathbf{t}_{\overline{A}}\right)^{-1}}. \tag{4.12}$$

Figure 4.4b illustrates this concept of the changes of body-fixed frame.

Another nice property of such a direct-product operation is that rotation and translation components of a change of poses can be split, thereby allowing individual considerations during a robotic task. Motivated by this spirit, the quantization for spatial motions can be separated into discussions of pure rotations and translations respectively.

4.3.2.2 Decoding algorithm for $\mathbf{SO}(3)$

With the separation of rigid-body motions into rotation and translation parts, the following contexts describe in details how the signal-to-symbol problem can be solved in the purely rotational case. Since Eq. (4.3) is a set rather than a group, it can be viewed as either a subset of $\mathrm{SE}(3)$ or $\mathrm{PCG}(3)$. Either way, the general decoding problem reduces to this: Given $H, K < \mathrm{SO}(3)$ and $R \in \mathrm{SO}(3)$, how to efficiently find the unique pair $(h_i, k_j) \in H \times K$ such that

$$R = h_i Q k_j \tag{4.13}$$

with $Q \in F_{H\backslash \mathrm{SO}(3)/K}$. Especially if the Voronoi choice is made for $F_{H\backslash \mathrm{SO}(3)/K}$, solving Eq. (4.13) allows for simply rounding off R to $h_i k_j$, as indicated in Sec. 3.1.2.

With the crystallographic constraint, in $\mathrm{SE}(3)$, it is possible to define H such that $|H| = 24$ (octahedral symmetry) and $|K| = 60$ (icosahedral symmetry), leading to $24 \times 60 = 1440$ combinations. In $\mathrm{PCG}(3)$, on the other hand, subgroups need not be restricted to the crystallographic constraints, therefore, more rotational elements can be obtained. A brute-force way to search for h_i and k_j that minimizes $\rho(R, h_i k_j)$ is through two nested for loops over i and j, which results in a $O(|H||K|)$ complexity of computations. However, by adopting the nice Voronoi properties, Eq. (4.13) can be solved for $F_{H\backslash \mathrm{SO}(3)/K}$ much more efficiently as compared to the brute-force searching method.

Consider the double-coset space $H\backslash \mathrm{SO}(3)/K$, where H is the group of rotational symmetries of the icosahedron, and $K = gHg^\top$ is a conjugated group with g being chosen so that $H \cap K = \{\mathbb{I}\}$. Then $|H\backslash \mathrm{SO}(3)/K| =$

$|H| \times |K| = 60^2 = 3600$. The fundamental domain $F_{H\backslash \mathrm{SO}(3)}$ for the coset space $H\backslash \mathrm{SO}(3)$ can be constructed as a dodecahedral Voronoi cell (i.e. as in Eq. (3.18) and Fig. 3.1c). Due to the Voronoi property, the shifted tile $h_i F_{H\backslash \mathrm{SO}(3)}$ containing the rotation R of interest can be computed using the distance $\rho(R, h_i)$ of R with the 60 tile centers $h_i \in H$. Then, $h_i^\top R$ lies in the identity-centered tile $F_{H\backslash \mathrm{SO}(3)}$. The fundamental domain $F_{H\backslash \mathrm{SO}(3)/K}$ can be constructed for the double-coset space as a Voronoi cell, too (i.e. as in Eq. (3.19) and Fig. 3.2). Since the shifts $h_i F_{H\backslash \mathrm{SO}(3)/K} k_j$ of this identity-centered fundamental domain will cover the whole space of $\mathrm{SO}(3)$, they will also cover $F_{H\backslash \mathrm{SO}(3)}$. But the number of required shifts will be much smaller than 3600. In this case, one can quickly find the shifted fundamental domain $h_{i'} F_{H\backslash \mathrm{SO}(3)/K} k_j$ that contains the above $h_i^\top R$ by exploiting the Voronoi property of these shifted domains. Then, $R \in (h_i h_{i'}) F_{H\backslash \mathrm{SO}(3)/K} k_j$, and the decomposition (4.13) is found. Note that the computations of the distance metric between two rotations uses the facts that: (1) $\rho(R_1, R_2) = \arccos(\frac{1}{2}[\mathrm{tr}(R_1^\top R_2) - 1])$, the trace wherein is computed as

$$\mathrm{tr}(R_1^\top R_2) \;=\; \sum_{i=1}^{3} \sum_{j=1}^{3} [R_1]_{ij} [R_2]_{ij} \,;$$

as well as (2) $\arccos x < \arccos y$ if and only if $x > y$.

Another case of decomposition is to consider the double-coset space $H\backslash \mathrm{SO}(3)/H$, where H is the group of rotational icosahedral symmetry. Here, the dodecahedral Voronoi fundamental domain $F_{H\backslash \mathrm{SO}(3)}$ is used for the coset space $H\backslash \mathrm{SO}(3)$, and the shifted tile $h_i F_{H\backslash \mathrm{SO}(3)}$ can be found that contains the rotation R easily as described above. For the fundamental domain $F_{H\backslash \mathrm{SO}(3)/H}$,

the tetrahedral wedge is chosen as shown in Fig. 3.3. The conjugated wedge $h_j F_{H\backslash \mathrm{SO}(3)/H} h_j^\top$ is found that contains the pulled-back rotation $h_i^\top R$ by using the method of querying a 3D point inside the tetrahedral simplex. Then, $R \in (h_i h_j) F_{H\backslash \mathrm{SO}(3)/H} h_j^\top$, which solves Eq. (4.13).

Note that as is the case in the previous Sec. 4.2, the shifts of the fundamental domains $F_{H\backslash \mathrm{SO}(3)/K}$ and $F_{H\backslash \mathrm{SO}(3)/H}$ above all have the same volume, which is an important advantage of the double-coset approach presented in this chapter.

4.4 Comparisons and Applications

Discretization on SO(3) is an important application of this work, which gives a equi-spaced decomposition of the group in the sense that any rotation locates inside an identical Voronoi cell. To clearly demonstrate the advantageous potential of the proposed discretization and decoding algorithms, benchmarks of performance with existing methods are conducted. As an application for this quantization work, a hybrid nearest neighbor searching algorithm is introduced, which is able to take the advantage of both the speed and low dispersion properties.

4.4.1 The Accuracy and Speed of Rounding Off Motions

4.4.1.1 Quality Measurements of Sampling $\mathbf{SO}(3)$: Discrepancy, Dispersion, Consistency and Uniformity

Several measures of the quality of a finite sampled set rotations have been proposed in the literature. One is *discrepancy* as defined and used in [23], [105],

[167]:

$$D(P,\mathcal{R}) := \sup_{R\in\mathcal{R}} \left| \frac{|P\cap R|}{|P|} - \frac{Vol(R)}{Vol(\mathrm{SO}(3))} \right|, \tag{4.14}$$

where $\mathcal{R}$ is a collection of measurable subsets of $\mathrm{SO}(3)$ and P is a finite set of sample points in SO(3), and $|S|$ is the number of elements in the finite set S. This concept was updated recently to include products of motion groups [9].

Given any such metric, $\rho : \mathrm{SO}(3)\times\mathrm{SO}(3) \rightarrow \mathbb{R}_{\geq 0}$, the *dispersion* of the points can be computed as

$$\mathfrak{D}(S,\rho) := \max_{R\in SO(3)} \left[\min_{R_S\in S} \rho(R,R_S) \right], \tag{4.15}$$

where S is a set of the sampled elements in $\mathrm{SO}(3)$.

In addition to discrepancy and dispersion, many other measures of sample quality can be defined. For example, a sampling can be called *consistent* if the distribution of the round-off error for any random rotation is concentrated. Therefore, the consistency of a set of samples on $\mathrm{SO}(3)$ can be defined as

$$\mathfrak{C}(S,\rho) := \sigma_{R\in S}\left\{m(R)\right\}, \tag{4.16}$$

where

$$m(R) := \min_{R_S\in S-\{R\}} \rho(R,R_S)$$

and σ is the standard deviation of the set of distances between each sample point and its nearest neighbor. Here a low value of $\mathfrak{C}(S,\rho)$ indicates high consistency. For example, if $\mathfrak{C}(S,\rho)$ is zero, then every point in the set has nearest neighbors of the same distance. Moreover, if each point in the set has many neighbors that achieve the minimal value of distance, then the set S has a high level of *uniformity*, which can be quantified as follows. For each $R\in S$

compute the subset $A_R \subset S$ as

$$A_R := \{Q \in S \,|\, \rho(R, Q) = m(R)\}.$$

Then the cardinality of this set measures the number of equally close nearest neighbors to R, and uniformity weights this by the spread of this number:

$$\mathfrak{U}(S, \rho) := \frac{\min_{R \in S} |A_R|}{1 + \max_{R \in S} |A_R| - \min_{R \in S} |A_R|}, \tag{4.17}$$

where a high value indicates a high uniformity. The numerator reflects the number of nearest neighbors for the worst sample, and the denominator reflects the spread (with 1 included since the difference between max and min can be zero).

For example, an integer lattice in 3D Euclidean space has a high level of uniformity with regard to the Euclidean metric, with each point having six nearest neighbors, each with the same minimized value of distance and $\mathfrak{C}(S, \rho) = 0$ and $\mathfrak{U}(S, \rho) = 6$. A spherical close-packing can have an even higher value of $\mathfrak{U}(S, \rho)$.

4.4.1.2 Uniformity of Sampling on SO(3)

For the evaluations of the proposed $\mathrm{SO}(3)$ discretization algorithm, three measures are compared: dispersion (Eq. (4.15)), consistency (Eq. (4.16)) and uniformity (Eq. (4.17)). Two other existing algorithms are considered: Euler angles and Hopf fibration [167]. For the computations of dispersion and consistency in this work,

$$\rho(R, R_s) := \| \log^{\vee}(R^{\top} R_S) \|_2$$

is the distance metric defined in Sec. 3.1.2. For the dispersion comparison, 10000 rotations are randomly generated and the distances with their nearest samples are computed. The maximum value of these 10000 resulting distances approximate the dispersion. For the consistency, on the other hand, for each sample, the distance to its nearest sample is computed and standard deviation measures the consistency. And for the uniformity measure, the number of nearest neighbors of each sample is computed.

For the proposed algorithm, the double-coset space $H\backslash \mathrm{SO}(3)/K$ is used, where H is the group of rotational symmetries of the icosahedron, and $K = gHg^{-1}$ where g is chosen such that

$$\log^{\vee} g \coloneqq [0.4359, -0.07692, -0.1282]^{\top}.$$

For the discretization using Euler angles, ZYZ parameterization is chosen, and α and γ are uniformly generated within their range $[-\pi, \pi]$ and β is generated such that $\cos\beta \in [-1, 1]$.

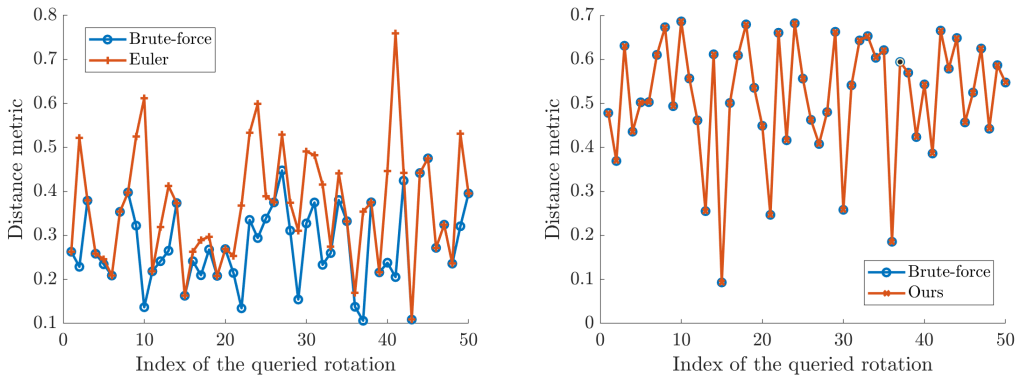
In terms of the dispersion, the proposed algorithm is higher than the other two: the dispersion of ours is 0.4291, Hopf fibration method is 0.2690 and Euler angle method is 0.3401. But for the consistency, ours can achieve 0 deviation, while Hopf is 0.0264 and Euler angle is 0.0865. This consistency result is a significant advantage of the proposed method in the sense that the sampling grid always has equi-distance edges. Also since the number of nearest neighbors with minimum distance for our method is always 2, the uniformity is 2. This also outperforms the other two methods, where Hopf fibration has a uniformity of 0.25, and Euler angle is only 0.0303. The results

show that our method can be used in uniformly sampling rotations of from $\mathrm{SO}(3)$.

4.4.1.3 Computational Time of SO(3) Nearest Neighbor Search

Another key factor for performance evaluation is the running time when searching for the nearest sample for a random rotation. A common and efficient way is using the Euler angle parameterization. Suppose a set of sample points constructed using Euler angles is given (either with or without $\cos^{-1}$ sampling, or Lattman's diagonalization of the metric tensor). Then given an arbitrary rotation, R, one can compute its Euler angles $(\alpha_R, \beta_R, \gamma_R)$ and attempt to round off to the nearest Euler angles in the sample set as $([\alpha_R], [\beta_R], [\gamma_R])$. This is simple and fast to compute, but because Euler angles are not an equi-metric spacing, the resulting rounded rotation matrix $[R] = R_3([\alpha_R])R_1([\beta_R])R_3([\gamma_R])$ may not be the closest of the sampled rotations to R. In contrast, the proposed coarse-to-fine decoding algorithm is both fast and accurate in the sense of metric round-offs.

To illustrate this, a benchmark is ran at an Intel Core i7-4790 CPU at 3.60 GHz and with Matlab R2018b. For the proposed method, the double-coset space $H\backslash\mathrm{SO}(3)/H$ is used. Comparisons are performed with the Euler angle seaching method (described above). 1000 random rotations are generated and localized to the nearest sample from the sampling list. The accuracy of computing the nearest neighbor is evaluated using the brute-force nearest neighbor search (i.e. minimization of the distance $\rho(R, h_i h_j)$ with respect to all i and j).



(a) Searches using Euler angles

(b) Searches using our method

Figure 4.5: Comparisons of the minimum distance between the queried rotation to the set of samples. The true values are computed using the brute-force nearest neighbor search, which is shown in blue curve. The Euler angle search sometimes returns higher values of distance; but ours can always give the correct answer.

The result shows that the proposed decoding algorithm yields an average runtime for each search at around 55.9μs for the proposed method, while the Euler angle method runs 53.6μs per rotation. Both methods are in the same level of efficiency, but ours outperforms in terms of round-off accuracy. Figure 4.5 shows the minimum distance between the queried rotations (50 of all the testing rotations are shown for a clearer plot) to the set of samples. Ours can always find the true nearest neighbor, while Euler angle sometimes return the sample which is not the nearest to the queried rotation.

4.4.2 Combining the Benefits of Speed and Good Dispersion Properties

Sampling methods such as those based on the Hopf fibration were designed for good performance in terms of minimal dispersion and consequently outperform both Euler angles and our proposed sampling algorithm in terms of

minimizing dispersion. Ours was designed for rapid query. However, if one wants both, it is possible to combine simply in a natural way as follows:

- Partition any given set of sample rotations with desirable properties (such as dispersion, discrepancy, etc.) by determining in which shard each sample point belongs;
- Given an arbitrary rotation, determine in which double-coset fundamental domain it belongs;
- Compute the distance between the arbitrary given rotation and all sample points in the same shard, and the those in nearest surrounding shards.

In this process, the number of sample points can be on the same order or even much higher than the number of shards.

Numerical simulations are conducted to verify this proposed hybrid searching algorithm using the $H\backslash\mathrm{SO}(3)/H$ decomposition. As a pre-processing step, the nearest neighbors for each of the 60 elements in H are computed, and each rotation is found to be surrounded by 12 neighbors. This step is to construct a connectivity map for the single-coset space, or in other words for each icosahedral cell. Then 10000 rotations are sampled using Hopf fibration, which is known to have low dispersion and discrepancy. Each sample is decomposed using the proposed coarse-to-fine decoding algorithm, whose index pair is stored in a pre-computed list that locates its corresponding cell and shard. Afterwards, 1000 random rotations are generated to be queried

to their closest sampling rotations. Each of these random rotations is decomposed and located into the cell (determined by the first index), then calculated the minimum distance with all the samples that locates in the same cell and the 12 neighboring cells. By using this hybrid method, the running time is around 5 times faster than the brute-force searching method, and the resulting minimum distance is verified to be 100% correct.

Another simulation is conducted using the $H\backslash\mathrm{SO}(3)/K$ decomposition. The difference with the previous test is the pre-process, where in this case, the nearest neighbors for the 3600 elements are computed, *i.e.*, $h_i k_j \in H \times K$ where $i, j = 1, 2, ..., 60$. This is equivalent to find the neighbors of each shard in the double-coset space. The same sampling and testing sets are input into the hybrid algorithm. The proposed hybrid algorithm can achieve an around of 20 times of speedup than the brute-force method. Also, the resulting minimum distance is verified as 100% accurate.

4.5 Chapter Summary

The quantization of continuous motions is introduced in this chapter via a class of motion alphabets. With such an alphabet, continuous motion trajectories can be captured with finite words/sentences. It is demonstrated in some examples how the possibility to construct fundamental domains for coset and double-coset spaces as Voronoi or Voronoi-like cells can be used to solve this decoding or signal-to-symbol problem efficiently via a coarse-to-fine searching algorithm. The performance, such as uniformity, of the proposed group discretization algorithm is compared with other two existing sampling

algorithms. And the running speed of the decoding algorithm is benchmarked with one other existing algorithm. The alphabets developed here can be used in the future to facilitate the connection between advances in artificial intelligence (such as the use of artificial neural networks) and physical robots acting in the world.

Chapter 5

Calibrating Rigid Transformations using Probability Densities on Euclidean Motion Groups

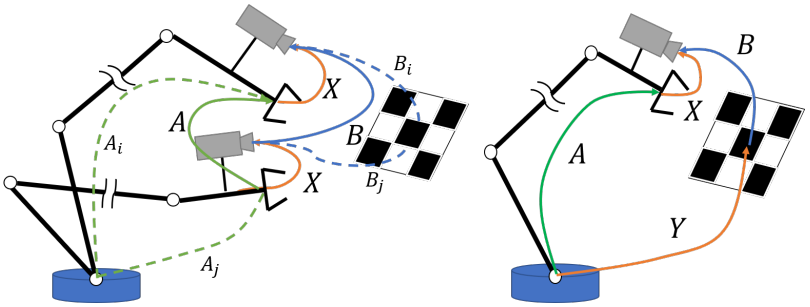
This chapter introduces a novel iterative algorithm to solve for a multi-robot calibration problem, i.e. $AXB = YCZ$. Here, A, B, C are rigid-body transformation data stream from sensor measurements, whereas X, Y, Z are the fixed unknown rigid transformations to be computed. This work builds on top of the previous probabilistic algorithms, which assumed the data acquired from sensors being lack of temporal correspondence. This assumption provides another type of calibration settings, where there might be scrambled or missing data from sensor readings. For example, the sampling frequencies of different sensors might be different, therefore, traditional calibration algorithms need careful pre-processes like time stamping. But with the spirit of probabilities on the group of rigid-body motions, these pre-processes can be simplified in the sense that only a distribution

of the data points are required. Apart from the ability to deal with non-correspondent data, this new algorithm further elevates the performance of the previous methods in dealing with noise introduced by the sensor readings. The codes that associated to algorithm implementations and performance benchmark in simulations and physical experiments are available in https://github.com/ruansp/axbycz_calibration.git.

5.1 Introduction

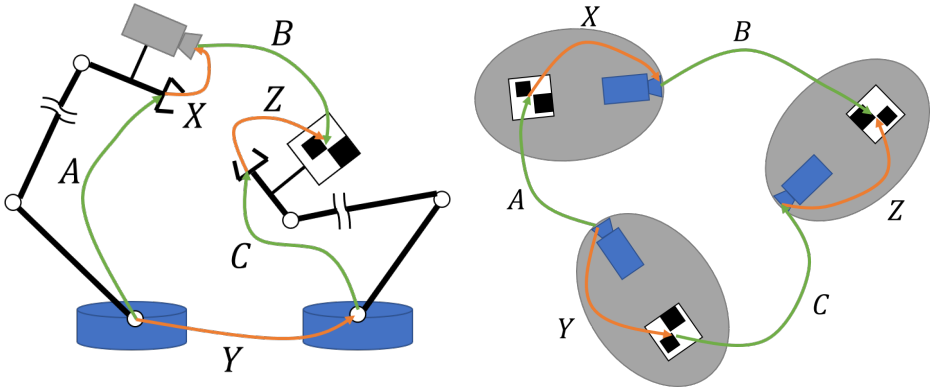
The most commonly studied extrinsic sensor calibration problems are the $AX = XB$ and $AX = YB$ problems, both of which uses only one robot with a camera mounted on its "wrist". The camera is able to capture the patterns of a checker board placed arbitrarily in the environment. The $AX = XB$ loop requires the robot to move into two different configurations and compute a relative transformation for the poses of end effector and the camera. In addition, the $AX = YB$ loop also involves the relative pose of the base frame of the robot. As an extension to these problems (as shown in Fig. 5.1), a multi-robot calibration problem can be formulated using the equation $AXB = YCZ$. Here, all the symbols represent the rigid-body transformations as elements in $\mathrm{SE}(3)$, where A, B, C are time-varying data stream from sensor measurements and X, Y, Z are fixed transforms to be solved.

Figure 5.2 demonstrates two typical systems for the multi-robot calibration problem. In particular, Fig. 5.2a shows a dual-arm system, which can calibrate the relative configurations between hand-eye, robot-robot and tool-flange. The poses of end effectors of the two robots can be obtained from the kinematic



(a) $AX = XB$ problem. **(b)** $AX = YB$ problem.

Figure 5.1: Demonstration of two types of extrinsic sensor calibration problems, i.e. $AX = XB$ hand-eye problem and $AX = YB$ hand-eye, robot-world problem.



(a) A dual-arm system. **(b)** A triple-mobile-robot system.

Figure 5.2: Demonstration of two typical systems for the $AXB = YCZ$ calibration problem.

information, and the camera mounted on one robot can see the tag mounted on the other. And Fig. 5.2b shows a group of three mobile robots, each of which has a camera (that can see the other robot) and a tag (that indicates the body frame). The goal is to find the relative configuration for the camera relative to the base frame of each robot.

Existing methods assume that the measured data is fully correspondent, therefore, the unknown transforms can be computed simultaneously [149],

[157], [159]. In particular, [149] presented an optimization method for the calibration of a cooperative dual-arm system, where one robot holds the camera and the other holds the marker; and [159] proposed optimization and Kronecker-product based approaches toward calibrating a serial-parallel manipulator. Recently, several probabilistic algorithms were presented and verified in simulations to be effective for data without correspondence, i.e. the ordering of data is scrambled or there are some missing data points [89], [101], [102]. These probabilistic solvers were presented based on the histograms of the measured dataset, which are normalized to be probability densities. As a novel addition, this chapter introduces an iterative algorithm to add robustness to measurement noise in experimental data. The approach is simple to implement and effective in handling noisy data when there is partial knowledge of the correspondence.

The rest of this chapter is organized as follows. Section 5.2 formulates the calibration problem formally and derive the fundamental mathematical equations. Section 5.3 reviews the previous probabilistic algorithms. Section 5.4 develops the proposed iterative refinement algorithm using the whole set of data. In Sec. 5.5, the proposed iterative refinement algorithm is compared in simulations with a traditional algorithm that assumes full data correspondence and the previous probabilistic algorithm. In Sec. 5.6, physical experiments are conducted using two NAO humanoid robots. The comparisons with existing algorithms show that the proposed iterative solver is able to further minimize the calibration errors in real settings. Finally, Sec. 5.7 draws conclusions and points out future directions.

5.2 Problem Formulation

This section formulates the mathematical foundations of the $AXB = YCZ$ problem. The two core equations to be solved are derived using the important properties of convolutions and probability density functions in $\mathrm{SE}(3)$. Then, the decompositions of these equations are computed in details.

5.2.1 Derivations of the core equations

The time-dependent data stream containing n points from the sensor measurements is denoted as the triplet $(A_i, B_i, C_i) \in SE(3) \times SE(3) \times SE(3)$ where $i = 1, \cdots, n$. Then, the equation of each correspondent data point can be expressed in the form of

$$A_i X B_i = Y C_i Z. \tag{5.1}$$

For an element $H \in \mathrm{SE}(3)$, the shifting property of Dirac delta function gives

$$\begin{aligned} (\delta_{A_i} * \delta_X * \delta_{B_i})(H) &= \delta(B_i^{-1} X^{-1} A_i^{-1} H) \\ (\delta_Y * \delta_{C_i} * \delta_Z)(H) &= \delta(Z^{-1} C_i^{-1} Y^{-1} H). \end{aligned} \tag{5.2}$$

The above two equations can be combined to replace Eq. (5.1) as

$$(\delta_{A_i} * \delta_X * \delta_{B_i})(H) = (\delta_Y * \delta_{C_i} * \delta_Z)(H). \tag{5.3}$$

By converting the original equation into this convolution form, all the indexed values can be summed since Eq. (5.3) is a scalar function. Meanwhile, it is also allowed to use the covariance propagation formulas introduced in Sec. 3.1.3. To solve for these equations, different types of constraints can be

introduced to the dataset $\{A_i\}, \{B_i\}, \{C_i\}$, by fixing one data stream while letting the other two vary. For example:

- for a system with three mobile robots, any two robot agents (i.e. A, B) can remain static while the third agent (i.e. C) moving freely; and
- for a dual-arm and serial-parallel robotic systems, A or C can be fixed individually while the other two will be moving.

Note that in the latter case, B describes the transformation between the marker frame and the camera frame, while A and C are solely determined using the forward kinematics of the robots. Hence it is very difficult to keep B constant while varying A and C. No matter which frame is fixed, the relationship in Eq. (5.3) still holds. Here, only the derivations when fixing A are provided. The cases when fixing B or C follow a similar spirit, and the results for all three cases are summarized later in Tab. 5.1.

If A_i is fixed to be a constant transformation A, then Eq. (5.3) becomes

$$(\delta_A * \delta_X * \delta_{B_i})(H) = (\delta_Y * \delta_{C_i} * \delta_Z)(H). \tag{5.4}$$

If n instances of Eq. (5.4) are considered and summed together, the convolutions of probability density functions can be obtained as

$$\frac{1}{n}\sum_{i=1}^{n}(\delta_A * \delta_X * \delta_{B_i})(H) = \frac{1}{n}\sum_{i=1}^{n}(\delta_Y * \delta_{C_i} * \delta_Z)(H).$$

The bi-linearity property of convolutions gives

$$\left(\delta_A * \delta_X * \frac{1}{n}\sum_{i=1}^{n}\delta_{B_i}\right)(H) = \left(\delta_Y * \frac{1}{n}\sum_{i=1}^{n}\delta_{C_i} * \delta_Z\right)(H).$$

Then,

$$(\delta_A * \delta_X * f_B)(H) = (\delta_Y * f_C * \delta_Z)(H). \tag{5.5}$$

The mean equation of $AXB = YCZ$ can be computed as

$$M_A M_X M_B = M_Y M_C M_Z. \tag{5.6}$$

Because X, Y, Z and A are all single elements of $SE(3)$, then $M_X = X$, $M_Y = Y$, $M_Z = Z$, $M_A = A$ and $\Sigma_X = \mathbb{O}$, $\Sigma_Y = \mathbb{O}$, $\Sigma_Z = \mathbb{O}$, $\Sigma_A = \mathbb{O}$. Therefore, Eq. (5.6) becomes

$$\boxed{AXM_B = YM_CZ} \tag{5.7}$$

The covariance Σ_{A*X*B} is obtained by the three-fold convolution as

$$\Sigma_{A*X*B} = Ad(B^{-1})Ad(X^{-1})\Sigma_A Ad^\top(X^{-1})Ad^\top(B^{-1}) + \Sigma_B = \Sigma_B. \tag{5.8}$$

Similarly, Σ_{Y*C*Z} can be obtained as

$$\Sigma_{Y*C*Z} = Ad(Z^{-1})\,\Sigma_C\, Ad^\top(Z^{-1}). \tag{5.9}$$

Therefore, by equating Eq. (5.8) and Eq. (5.9), the covariance equation for $AXB = YCZ$ with A fixed becomes

$$\boxed{\Sigma_B = Ad(Z^{-1})\,\Sigma_C\, Ad^\top(Z^{-1})}. \tag{5.10}$$

Eq. (5.7) and Eq. (5.10) are the two fundamental equations to be solved for the unknown transformations X, Y, Z.

5.2.2 Decomposition of covariance equations

In order to decompose Eq. (5.10), the covariance matrix can be defined as

$$\Sigma_H = \begin{bmatrix} \Sigma_H^1 & \Sigma_H^2 \\ \Sigma_H^3 & \Sigma_H^4 \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{5.11}$$

where $H = A, B, C$ and $\Sigma_H^i \in \mathbb{R}^{3\times 3}$. Substituting Eq. (5.11) into Eq. (5.10) gives the upper left block as

$$\Sigma_B^1 = R_Z^\top \Sigma_C^1 R_Z, \tag{5.12}$$

and the upper right block as

$$\Sigma_B^2 = R_Z^\top \Sigma_C^1 R_Z \widehat{\boldsymbol{t}}_Z^\top + R_Z^\top \Sigma_C^2 R_Z \tag{5.13}$$

To make clear representations, Eq. (5.12) is denoted as the *Sig-Rot* equation (i.e. containing rotation part only) and Eq. (5.13) is denoted as the *Sig-Trans* equation (i.e. containing translation part only).

With only these two equations, it is not sufficient to solve the problem. This is because Eq. (5.12) contains only R_X and R_Z, and Eq. (5.13) contains only $\boldsymbol{t}_X$ and $\boldsymbol{t}_Z$, whereas the information regarding Y is not available to be achieved. Therefore, some rearrangements of the order of X, Y, Z are required by fixing different data streams (i.e. A, B, C). There are a total of six variations of $AXB = YCZ$ formulations by pre-/post-multiply different transformation matrices. Table 5.1 summarizes these six variances as well as their correspondent Sig-Rot equations.

No.	Representation	Fixed	Sig-Rot
1	$AXB = YCZ$	A	$\Sigma_B^1 = R_Z^T \Sigma_C^1 R_Z$
2	$A^{-1}YC = XBZ^{-1}$		
3	$BZ^{-1}C^{-1} = X^{-1}A^{-1}Y$	B	$R_C \Sigma_C^1 R_C^T = R_Y^T R_A \Sigma_A^1 R_A^T R_Y$
4	$B^{-1}X^{-1}A^{-1} = Z^{-1}C^{-1}Y^{-1}$		
5	$C^{-1}Y^{-1}A = ZB^{-1}X^{-1}$	C	$R_B \Sigma_B^1 R_B^T = R_X^T \Sigma_A^1 R_X$
6	$CZB^{-1} = Y^{-1}AX$		

Table 5.1: Variations of the calibration equations by fixing different data streams.

5.3 Review of Probabilistic Algorithms

This section reviews the previously proposed probabilistic algorithms from different settings of the problems. The two algorithms, i.e. *Prob1* and *Prob2*, fixes different sets of the data and recover rotation and translation parts of the unknown transformations separately.

5.3.1 Algorithm *Prob 1*: fixing A and C

In this algorithm, two representations from Tab. 5.1 are chosen, i.e. No. 1 and 6. For representation 1, A is fixed, or equivalently $A = A_I$, and datasets $\{B_{Ii}\}$ and $\{C_{Ij}\}$ are measured where $i, j = 1, \cdots, n$. The objective is to first solve for the rotation part using the Sig-Rot equation

$$\Sigma_{B_I}^1 = R_Z^\top \Sigma_{C_I}^1 R_Z. \tag{5.14}$$

Since Eq. (5.14) is a similarity transformation, Σ_B^1 and Σ_C^1 have the same eigenvalues. Calculating the eigenvalue decomposition of Σ_B and Σ_C gives

$$\Sigma_B^1 = Q_B \Lambda Q_B^\top \text{ , and } \Sigma_C^1 = Q_C \Lambda Q_C^\top, \tag{5.15}$$

where Λ denotes the diagonal matrix. Substituting Eq. (5.15) into Eq. (5.14) gives

$$\Lambda = Q_B^\top R_Z^\top Q_C \Lambda Q_C^\top R_Z Q_B := \mathcal{Q}\Lambda\mathcal{Q}^\top. \tag{5.16}$$

The special structure of Eq. (5.16) gives 4 solutions for $\mathcal{Q}$ [1], i.e.

$$\mathcal{Q} = \left\{ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \right\}. \tag{5.17}$$

Thus, the 4 candidates for R_Z can be computed as

$$\boxed{R_Z = Q_{C_I} \mathcal{Q} Q_{B_I}^\top}. \tag{5.18}$$

For the translation part $\boldsymbol{t}_Z$, Eq. (5.13) can be simplified as

$$\Sigma_{B_I}^2 = R_Z^\top \Sigma_{C_I}^1 R_Z \widehat{\boldsymbol{t}}_Z^\top + R_Z^\top \Sigma_{C_I}^2 R_Z \,, \tag{5.19}$$

where $\boldsymbol{t}_Z$ can be solved directly.

Similarly, when fixing $C = C_{II}$, its covariance and inverse are zero matrices. Therefore, the Sig-Rot equation of representation 6 is

$$\Sigma_{B_{II}^{-1}}^1 = R_X^\top \Sigma_{A_{II}}^1 R_X, \tag{5.20}$$

which has the similar structure with Eq. (5.16). Thus, R_X also has 4 candidates, which can be calculated as

$$\boxed{R_X = Q_{A_{II}} \mathcal{Q} Q_{B_{II}^{-1}}^\top}. \tag{5.21}$$

To recover Y, the mean equations using the candidates of X and Z are

employed as

$$\boxed{Y = A_I X M_{B_I} Z^{-1} M_{C_I}^{-1}}, \text{ and } \boxed{Y = M_{A_{II}} X M_{B_{II}} Z^{-1} C_{II}^{-1}}, \tag{5.22}$$

which gives $16 + 16 = 32$ candidates of Y. In total, there are $4 \times 4 \times 32 = 512$ candidate combinations of $\{X, Y, Z\}$. In order to find the best combination among the 512 choices, the two datasets can be used to minimize the error function as follows (for simplicity, let $M_{L_I} = A_I X_i M_{B_I}$, $M_{R_I} = Y_j M_{C_I} Z_k$, $M_{L_{II}} = M_{A_{II}} X_i M_{B_{II}}$ and $M_{R_{II}} = Y_j C_{II} Z_k$).

$$\begin{aligned} &\min || \log^{\vee}(R_{M_{L_I}}^{\top} R_{M_{R_I}})||_2 + || \log^{\vee}(R_{M_{L_{II}}}^{\top} R_{M_{R_{II}}})||_2 \\ &w \cdot ||\boldsymbol{t}_{M_{L_I}} - \boldsymbol{t}_{M_{R_I}}||_2 + w \cdot ||\boldsymbol{t}_{M_{L_{II}}} - \boldsymbol{t}_{M_{R_{II}}}||_2 \end{aligned} \tag{5.23}$$

where $i, j = 1, 2, 3, 4, k = 1, \ldots, 32$. Here w is the weighting factor and can be varied depending on the precision requirement on rotation and translation.

5.3.2 Algorithm *Prob 2*: fixing A, B and C

In addition to fixing A or C, the transformation B can also be fixed, which will produce three datasets that are labeled as follows:

Dataset I (fixing A): $A = A_I, \{B_{Ii}\}, \{C_{Ij}\}$; Dataset II: $B = B_{II}$ with $\{A_{IIi}\}$ and $\{C_{IIj}\}$; and Dataset III: $C = C_{III}$ with $\{A_{IIIi}\}$ and $\{B_{IIIj}\}$

Under this setting, X, Y, Z can be solved independently with each dataset and there are a total of $4 \times 4 \times 4 = 64$ combinations of solutions. To choose the best combination, the similar optimization with *Prob 1* is used. The object function can be defined as follows (letting $M_{L_{III}} = M_{A_{III}} X_i M_{B_{III}}$ and $M_{R_{III}} =$

$Y_j C_{III} Z_k$).

$$\begin{aligned} \min & || \log^{\vee}(R_{M_{L_I}}^{\top} R_{M_{R_I}})||_2 + || \log^{\vee}(R_{M_{L_{II}}}^{\top} R_{M_{R_{II}}})||_2 \\ & || \log^{\vee}(R_{M_{L_{III}}}^{\top} R_{M_{R_{III}}})||_2 + w \cdot ||\boldsymbol{t}_{M_{L_I}} - \boldsymbol{t}_{M_{R_I}}||_2 + \\ & w \cdot ||\boldsymbol{t}_{M_{L_{II}}} - \boldsymbol{t}_{M_{R_{II}}}||_2 + w \cdot ||\boldsymbol{t}_{M_{L_{III}}} - \boldsymbol{t}_{M_{R_{III}}}||_2, \end{aligned} \tag{5.24}$$

where $i, j, k = 1, 2, 3, 4$.

5.4 An Iterative Algorithm for Combined Dataset

This section proposes a novel iterative algorithm to solve for the equations in Sec. 5.2, where the input is a combined dataset of different configurations of fixed frames. More specifically, the algorithm mainly deal with the case when frame A or frame C is fixed at different poses, indexed by i and j respectively, and the other two frames are measured for each fixed value of A_i or C_j.

The equations to be solved are chosen from Tab. 5.1 with the representations and the corresponding covariance, where the full covariance matrices are used instead of the Sig-Rot part only. This algorithm is able to recover the rotation and translation parts of the unknown matrices simultaneously.

Explicitly, when A_i is fixed, Representation 1 is chosen and the corresponding equation for the covariance matrices is

$$\begin{cases} A_i X M_{B_i} = Y M_{C_i} Z \\ \\ \Sigma_{B_i} = Ad(Z^{-1}) \Sigma_{C_i} Ad^{\top}(Z^{-1}), \end{cases} \tag{5.25}$$

where A_i is the i^{th} pose when frame A is fixed, M_{B_i} and M_{C_i} are the mean of

the sets $\{B_i\}$ and $\{C_i\}$ when A_i is fixed at the i^{th} pose, Σ_{B_i} and Σ_{C_i} are the covariance of $\{B_i\}$ and $\{C_i\}$, $Ad(Z^{-1})$ is the adjoint of Z^{-1}.

As C_j is fixed, on the other hand, Representation 6 is picked and its corresponding equation for the covariance matrices is

$$\begin{cases} C_j Z M_{B_j}^{-1} = Y^{-1} M_{A_j} X \\ \\ \Sigma_{B_j^{-1}} = Ad(X^{-1}) \Sigma_{A_j} Ad^{\top}(X^{-1}), \end{cases} \tag{5.26}$$

where C_j is the j^{th} pose when frame C is fixed, M_{A_j} and M_{B_j} are the mean of the sets $\{A_j\}$ and $\{B_j\}$ when C_j is fixed at the j^{th} pose, Σ_{A_j} and $\Sigma_{B_j^{-1}}$ are the covariance of $\{A_j\}$ and $\{B_j^{-1}\}$, $Ad(X^{-1})$ is the adjoint of X^{-1}.

This system of equations (5.25) and (5.26) can be solved at the same time by adding small variations $\boldsymbol{\xi}_X$, $\boldsymbol{\xi}_Y$ and $\boldsymbol{\xi}_Z$ to X, Y and Z in the space of Lie algebra respectively. Using the first-order Taylor series approximation, the variables in the $k+1$ iteration can be computed as

$$\begin{aligned} X_{k+1} &= X_k(\mathbb{I} + \widehat{\boldsymbol{\xi}}_{X_k}) \\ \\ Y_{k+1} &= X_k(\mathbb{I} + \widehat{\boldsymbol{\xi}}_{Y_k}) \\ \\ Z_{k+1} &= X_k(\mathbb{I} + \widehat{\boldsymbol{\xi}}_{Z_k}), \end{aligned} \tag{5.27}$$

where k is the number of iterations. Note that these updating expressions will push the updated transformations $X_{k+1}, Y_{k+1}, Z_{k+1}$ off the Lie group of $SO(3)$ when the variances have large magnitude. Therefore, expressions in Eq. (5.27) are only used before solving for the unknowns $\boldsymbol{\xi}_{X_k}, \boldsymbol{\xi}_{Y_k}, \boldsymbol{\xi}_{Z_k}$. Afterwards, to keep the updated transformations within the group of $SO(3)$, the exact

exponential mapping is applied, i.e.

$$\begin{aligned} X_{k+1} &= X_k \exp\left(\widehat{\boldsymbol{\xi}}_{X_k}\right) \\ Y_{k+1} &= X_k \exp\left(\widehat{\boldsymbol{\xi}}_{Y_k}\right) \\ Z_{k+1} &= X_k \exp\left(\widehat{\boldsymbol{\xi}}_{Z_k}\right). \end{aligned} \tag{5.28}$$

To solve for the unknown variances at each iteration, Eqs (5.25) and (5.26) can be expanded by Eq. (5.27), which results in a linear system of equations

$$P_k \boldsymbol{\xi}_k = \begin{bmatrix} P_{1k} \\ P_{2k} \\ P_{3k} \\ P_{4k} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_{X_k} \\ \boldsymbol{\xi}_{Y_k} \\ \boldsymbol{\xi}_{Z_k} \end{bmatrix} = \boldsymbol{b}_k = \begin{bmatrix} \boldsymbol{b}_{1k} \\ \boldsymbol{b}_{2k} \\ \boldsymbol{b}_{3k} \\ \boldsymbol{b}_{4k} \end{bmatrix}, \tag{5.29}$$

where $\boldsymbol{\xi}_{X_k}, \boldsymbol{\xi}_{Y_k}, \boldsymbol{\xi}_{Z_k} \in \mathbb{R}^{6\times 1}$ are variables to be solved in each iteration; and the calculation of $P_{1k} \in \mathbb{R}^{12\times 18}$, $P_{2k} \in \mathbb{R}^{36\times 18}$, $P_{3k} \in \mathbb{R}^{12\times 18}$ and $P_{4k} \in \mathbb{R}^{36\times 18}$, and the corresponding $\boldsymbol{b}_{1k} \in \mathbb{R}^{12\times 1}$, $\boldsymbol{b}_{2k} \in \mathbb{R}^{36\times 1}$, $\boldsymbol{b}_{3k} \in \mathbb{R}^{12\times 1}$ and $\boldsymbol{b}_{4k} \in \mathbb{R}^{36\times 1}$ are shown in the Appendix B.2.

Once the variable $\boldsymbol{\xi}_k$ is solved by inverting P_k (the pseudo-inverse should be used here) to the right hand side, the matrices to be calibrated can be updated using Eq. (5.28). This process will converge to satisfy all the four equations where in the ideal case $\boldsymbol{\xi}_k = \mathbf{0}$.

5.5 Algorithm Benchmark in Simulation

To validate the performance, in this section, the proposed iterative refinement algorithm is benchmarked with the existing *Prob 1* and a traditional algorithm proposed by *Wang* [149] in simulation.

5.5.1 Data Generation

The ground truths for the unknowns are pre-defined. And according to the original calibration equation and the ground truths, the parameters A, B, C are generated as follows. Two subsets of data are collected for the simulations: fixing A and C respectively. When fixing A, a stream containing 100 elements of B is generated at random, which subject to Gaussian distributions around 5 different center poses. Then, the stream of C can be computed as

$$C_i = Y_{true}^{-1} A X_{true} B_i Z_{true}^{-1}, \tag{5.30}$$

where B_i and C_i are the i-th element in the stream, A is the fixed parameter whose index is dropped and $X_{true}, Y_{true}, Z_{true}$ are the ground truth values of the unknown transformations. Similarly, the data streams when fixing C can be also generated.

To simulate the measurement noise, variations are introduced for each generated data point. For example, B_i is firstly generated around a centered pose and subject to a Gaussian distribution, and also subject to Gaussian white noise defined as

$$B_i' = \exp(\widehat{\boldsymbol{\xi}}_{noise}) B_i, \tag{5.31}$$

where $\boldsymbol{\xi}_{noise} \sim \mathcal{N}(\mathbf{0}, \sigma_{noise}\mathbb{I})$ is a zero mean ($\mathbf{0} \in \mathbb{R}^6$) Gaussian with covariance being $\sigma_{noise}\mathbb{I} \in \mathbb{R}^{6\times 6}$. For this simulation, it is chosen as $\sigma_{noise} = 0.1$.

Note that for *Prob 1* algorithm, the two subsets are considered separately, while for the proposed iterative and Wang's algorithms, the combination of the whole data set is fed into the computations.

5.5.2 Error Metric

Since in simulation, ground truths are given a priori. The solved results can be compared with the true values of the unknowns. In this work, the error metric in simulation is computed by separating the rotation and translation parts between the solved transformation $H_{solved} = (R_{solved}, \mathrm{t}_{solved})$ and ground truths $H_{true} = (R_{true}, \mathrm{t}_{true})$, i.e.

$$\begin{aligned} \mathrm{Error}_R &= \left\| \log^{\vee}(R_{solved}^{\top} R_{true}) \right\|_2 , \\ \mathrm{Error}_{\mathbf{t}} &= \| \mathbf{t}_{solved} - \mathbf{t}_{true} \|_2 , \end{aligned} \tag{5.32}$$

where Error_R and $\mathrm{Error}_{\mathbf{t}}$ are the rotation and translation errors respectively.

5.5.3 Comparison Results and Analysis

In the simulation, 20 different triplets of the unknown transformations X, Y, Z are randomly chosen. And the dataset are generated using the process described above. At the mean time, a portion of the generated data is scrambled, ranging from 0 to 100%. The scrambling of the data means that the forms and sizes of data inputs are the same, but the orders are changed.

Figure 5.3 shows the comparison results of the calibration errors from the three benchmarked algorithms. Errors for the rotation and translation parts are separated computed and plotted accordingly. The figure shows the averaged errors for the 20 simulation trials.

All the three algorithms correctly return the calibration results when the data has exact correspondence. As the scramble rate becomes larger, the traditional algorithm has increasing amount of errors, but both the probabilistic

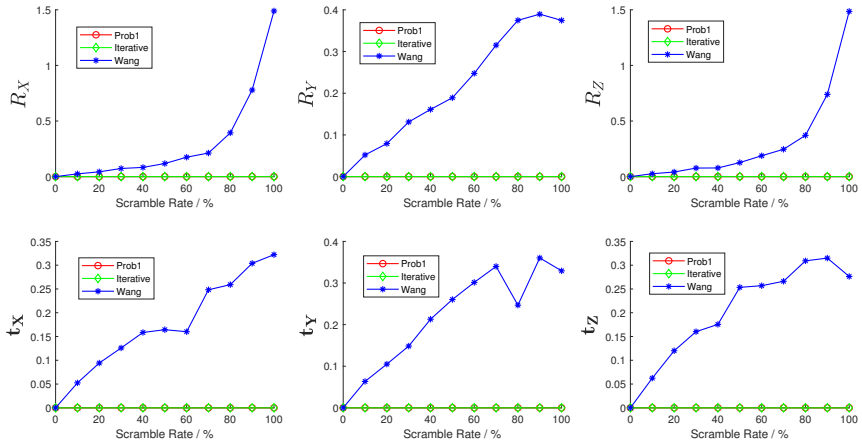


Figure 5.3: Simulation results of the calibration errors comparisons for the proposed iterative refinement, *Prob 1* and *Wang's* algorithms. The errors are computed based on the ground truths, with rotation and translation parts separated. Results from *Prob 1* is used as the initial guess of the proposed iterative refinement algorithm.

algorithms perform stable. This shows the advantage of these two probabilistic algorithms, which are robust no matter how disorder the data points are. Also, the results from the iterative refinement algorithm performs as good as *Prob 1*, which validates its effectiveness in solving the calibration problem in simulation. Since *Prob 1* already provides solutions with small amount of errors, the iterative refinement algorithm benefits from these initial conditions. Therefore, when it locally converges to an optimal solution using the full knowledge of the mean and covariance of the observed data, the optima stays very close to the initial condition, which keeps the calibration errors at a low level.

The next section will show some experimental validations for a real setting using two humanoid robots. The results will further show the advantages of the proposed iterative algorithm in a real robotic system, while the previous

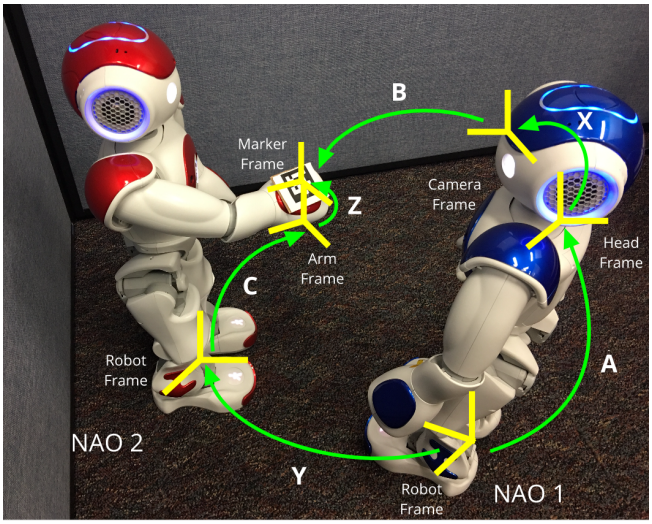


Figure 5.4: Real-world experiment settings for calibration using two NAO robots

probabilistic algorithm might drop the calibration accuracy.

5.6 Experimental Validation

In this section, physical experiments are conducted to further verify the extendability of the proposed iterative refinement algorithm. The platform is constructed by two NAO robots, where one is moving its hand and the other retrieves the relative transformations by the camera on the head. The data collected is put into both Wang's [149] approach and the previous probabilistic approach for comparisons.

5.6.1 Experiment Settings

As is shown in Fig. 5.4, the two NAO robots are set to stand and face to each other, and the description of transformation matrices are summarized in Tab. 5.2.

Transformation	Starting Frame	Ending Frame
A	Robot Frame of NAO 1	Head Frame of NAO 1
B	Camera Frame of NAO 1	Marker Frame on NAO 2
C	Robot Frame of NAO 2	Arm Frame of NAO 2
X	Head Frame of NAO 1	Camera Frame of NAO 1
Y	Robot Frame of NAO 1	Robot Frame of NAO 2
Z	Arm Frame of NAO 2	Marker Frame on NAO 2

Table 5.2: Summary of transformations that are measured and to be calibrated.

The transformations A and C can be measured using the subroutine in the SDK for programming provided by the company, and the transformation information of B is retrieved by attaching a marker on the right hand of NAO 2 and using the *ArUco* library [53].

The experiments are performed by first fixing the head of NAO 1 to a pose, and moving the arm of NAO 2, which is called a *trial* of data, followed by changing the head frame into several different poses and moving the arm of NAO 2 again. This gives several trials of data with fixed A and varying B and C. Trials of data with fixed C can be obtained by fixing the arm of NAO 2 to several different poses and moving the head of NAO 1. The current experimental data includes: (1) 3 different trials with fixed frame A and 50 sets of changing frames B and C; (2) 3 different trials with fixed frame C and 50 sets of changing frames A and B. In total, there are 300 pairs of measured data $\{A, B, C\}$ that are with correspondence. But since the camera cannot always detect and measure the transform of the *ArUco* marker, only 298 pairs of the data are valid without missing frames. Fig. 5.5 shows the sequence of moving the arm of NAO 2 while the head of NAO 1 is fixed in different poses.

The data are then labeled as $\{A_i, B_i, C_i\}$ where $i = 1, 2, 3$ for 3 trials of data

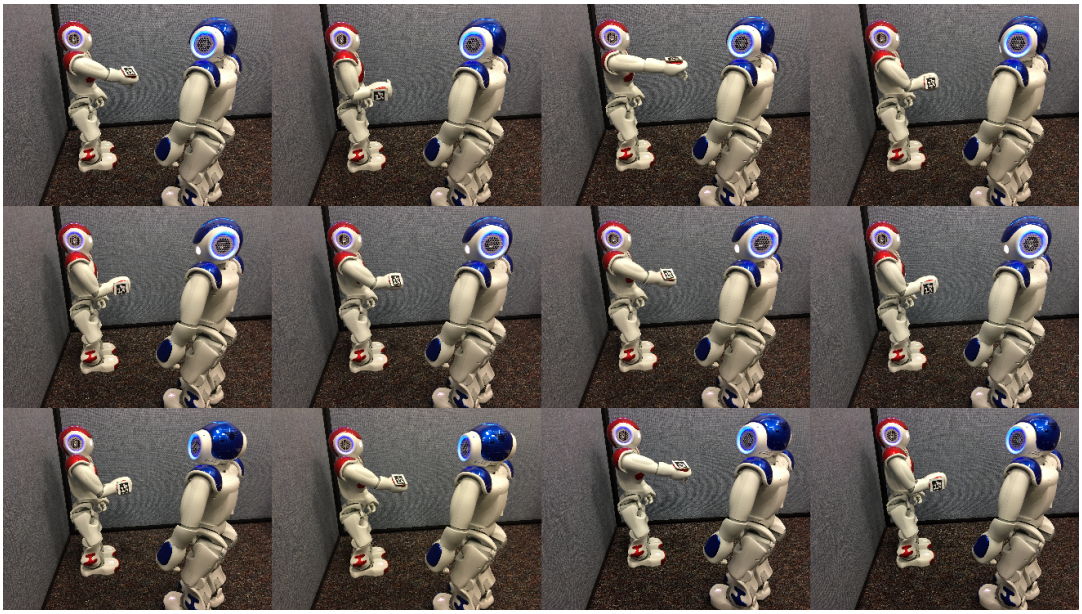


Figure 5.5: Moving sequence of the experiment.

with fixed A_i, and $\{A_j, B_j, C_j\}$ where $j = 1, 2, 3$ for 3 trials of data with fixed C_j,

5.6.2 Error Metric for Experimental Validation

Since for physical experiments, there are no ground truths for comparisons of calibrated transformations, the error metric can be computed between the left and right side of the calibration equation. In particular, the closeness between the left and right hand side of the basic equation $AXB = YCZ$ is computed, and the mean of the accumulated error for all the pairs of data is evaluated. In other words, the error metric for experimental data can be defined as

$$Error = \frac{1}{N}\sum_{m=1}^{N} \|A_m X_{solved} B_m - Y_{solved} C_m Z_{solved}\|_F \,, \tag{5.33}$$

Algorithm	Data Combinations
Wang	Combination of the whole dataset $\{A, B, C\}$;
Prob Iterative	Combined data: $\{A_i, B_i, C_i\}$ and $\{A_j, B_j, C_j\}$, where $i, j = \{1, 2, 3\}$.

Table 5.3: Summary of the data combination as inputs of each algorithm.

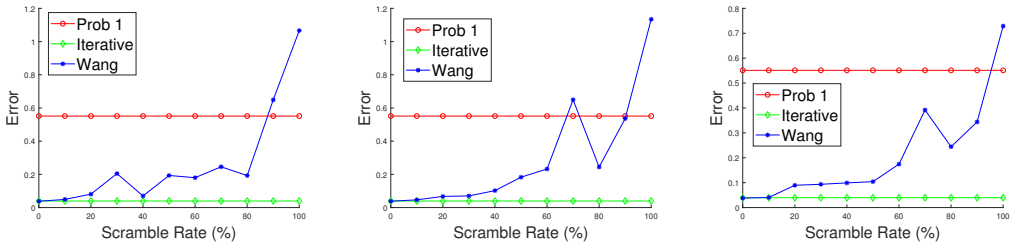
where the pair $\{A_m, B_m, C_m\}$ is the data with correspondence and N is the number of the pairs of data in the whole dataset.

5.6.3 Data Processing

Three different algorithms are compared: *Wang's*, *Prob 1* and the *iterative* refinement algorithms. The other version of probabilistic method, i.e. *Prob 2*, is not tested because it requires to fix B matrix. In the experimental settings of *Prob 2*, it is a nontrivial task to fix the transformation between camera and the marker. The data is proceeded in the following ways according to the requirements and assumptions of each algorithm:
(1) for *Wang's* method, all the data pairs are stacked together as $\{A, B, C\}$ where each of the matrices contains the whole sets of the transformations;
(2) for both the *Prob 1* and the proposed iterative algorithms, the data for fixed A and fixed C is separated, and treated as two sets of inputs.

Table 5.3 summarizes the combination of data that are put into those algorithms according to their requirements and assumptions.

To verify that the probabilistic methods can deal with the data without correspondence, the order of the data is scrambled and the errors of different methods are compared. Note that the scrambling of data is the same as



(a) From identity matrices. **(b)** From *Prob 1* results. **(c)** From measurement.

Figure 5.6: Errors with scrambling rate on real data with different initial guesses.

described in Sec. 5.5; and the error is still calculated using the data with correct correspondence.

5.6.4 Results and Analysis

The results for algorithm comparisons using experimental data with respect to the scramble rate are shown in Fig. 5.6, where the three plots show the error as the scrambling rate increases with different initial guesses.

As can be seen from the results, the errors using the probabilistic approach are invariant to the correspondence of the data, while using the traditional method the error increases significantly if the data is scrambled. In this aspect, once the collected data has some missing parts or the order is not correct by accident, the iterative refinement can still get a solution that is close to correct. Further, if the data is *rich* enough, the covariance of the dataset will become even more robust to recover the unknown transformations. Here rich means that the number of data collected is large, and the degrees of freedom for the moving part of both robots are high, which make all the measured transformations vary on a larger space so that the distribution can

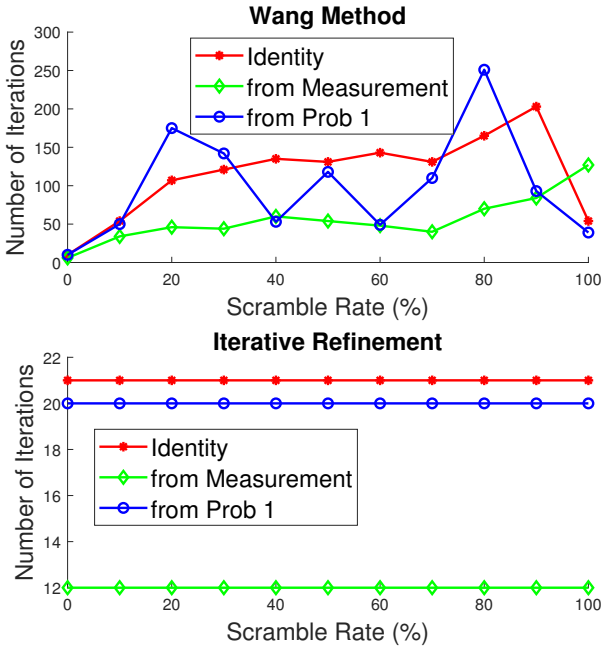


Figure 5.7: Number of iterations v.s. scrambling rate on experimental data with different initial guesses for *Wang's* and iterative algorithms.

be approximated closer to a Gaussian.

The initial guess also plays a role on the efficiency of the two algorithms. As shown in Fig. 5.7, the number of iterations differs with the changes of the initial guesses. The results from *Prob 1* can be a starting point to the iterative refinement, which gives faster convergence than an arbitrary guess, e.g. identity matrix. Further, a measurement of the transformations can be manually approximated, for instance, from the kinematic data of the robot, the algorithms can perform even more efficiently.

5.7 Chapter Summary

This chapter proposes a new iterative algorithm to solve for the $AXB = YCZ$ calibration problem with the lack of data correspondence. The core idea is based on the variation of the calibration equations by fixing different observation datasets. Two previous probabilistic algorithms are reviewed, whose results can be treated as initial conditions of the proposed iterative algorithm. The iterative algorithm uses a combined dataset when fixing either A or C, and solves for the rotation and translation parts of the unknown transformations simultaneously. Simulated as well as physical experimental results show that this algorithm is able to deal with data with large noise and scrambled data. This algorithm not only preserves the nice properties of the previous probabilistic algorithm, but also can deal with the real-world data which contains sensor noise. The experiments also show that the iterative refinement reliably converges to the correct answer if a good initial condition is provided.

Part III

Geometric Paradigm for Robot Motion Planning Based on Parameterizations of Free Space

Chapter 6

Collision Detection and Proximity Queries between Ellipsoids and Superquadrics

This chapter discusses the collision and proximity queries using the novel closed-form characterizations of Minkowski sums/differences between an ellipsoid and a general surface. The Minkowski sums and differences parameterize the boundary of contact space between two rigid bodies with fixed orientations, and is widely used in answering collision and distance queries for robots. Optimization-based algorithms are proposed and compared with the existing mesh-based algorithm.

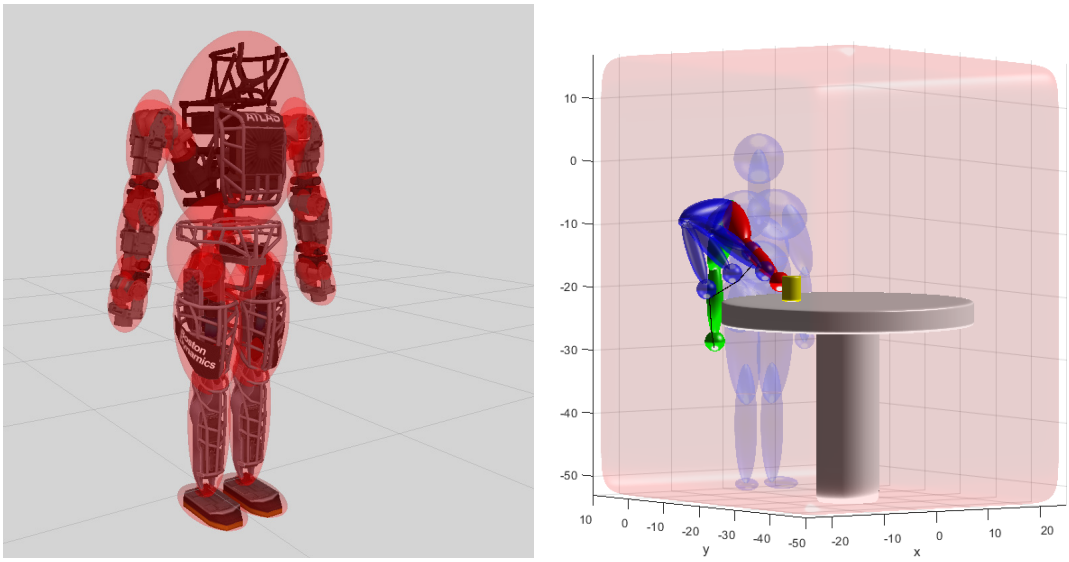
6.1 Introduction

Computing the distance between two rigid objects can be applied in many areas such as computer aided-design (CAD), robot motion planning and computer simulations. Many algorithms have been proposed to make the proximity queries more efficient when using polyhedral objects. However

these algorithms rely significantly on the complexity of the surface meshes bounding the objects, which results in a trade-off between accuracy and efficiency. Superquadrics, with ellipsoids being a simplified version within this family of shapes, have become popular recently since they require fewer representation parameters. A real-life scenario using superquadrics is a humanoid robot trying to pick up a cup on a table while avoiding hitting objects in its trajectory. The rigid parts of the robot are encapsulated by a union of ellipsoids and the objects in the environment are enclosed by superquadrics, as shown in Fig. 6.1. In [104], superquadrics are used for representing different objects in an environment where a PR2 has the task of grasping objects. The major advantage of superquadrics lies in the simple mathematical expressions and varieties of shapes it can describe without meshes. This section provides a new framework for collision detection and proximity queries between an ellipsoidal and a superquadric object based on the idea of closed-form Minkowski sums [164]. The major contributions are:

- An efficient algorithm for collision detection based on a parametric closed-form Minkowski sum expression is proposed;
- An optimization-based algorithm for proximity queries is proposed as an extension to the collision detection algorithm;
- An accuracy metric for discretization of surfaces is proposed for inexact algorithms using meshes, based on the Principal Kinematic Formula.

The rest of this chapter is organized as follows. Section 6.2, proposes an algorithm for collision detection between ellipsoids and superquadrics, both in



(a) An Atlas humanoid robot with rigid parts being encapsulated by ellipsoids.

(b) A poly-ellipsoidal humanoid robot picking up a cup while avoiding hitting the table.

Figure 6.1: An Examples of the scenarios where ellipsoids and superquadrics come to play a role in robot motion planning tasks.

2D and 3D. Section 6.3 provides solutions to the proximity queries between an ellipsoid and a superquadrics based on the closed-form Minkowski sums. And an accuracy metric of evaluations is introduced for mesh-based algorithms in Sec. 6.4. Details of benchmarks with the existing algorithms are provided in Sec. 6.5, and discuss and analyze the results in Sec. 6.6. Finally, Sec. 6.7 summarizes this chapter.

6.2 Collision Detection Algorithm Based on Closed-form Minkowski Sums

The parametric closed-form Minkowski sums computed in Eq. (3.51) shrink the moving ellipsoid to a point in $\mathbb{R}^n$. Therefore, the collision detection problem can be viewed as checking whether the point is outside of a parametric

surface.

6.2.1 Relative Position Between a Point and a Parametric Surface

Suppose the point to be checked is denoted as $\boldsymbol{x}_0 = [x_1, x_2, ..., x_n]^\top \in \mathbb{R}^n$, then the problem is to query the relative position between this point and a parametric surface S_a in $\mathbb{R}^n$. The idea is to firstly find a point $\boldsymbol{x}_{mb}$ on the parametric Minkowski sum boundary surface that falls on the line defined by origin O and the point $\boldsymbol{x}_0$, denoted as $l_{O\boldsymbol{x}_0}$. Then if $\boldsymbol{x}_0$ is farther from the origin than $\boldsymbol{x}_{mb}$, the point is outside of the boundary surface, therefore, the moving ellipsoid is separated from the fixed superquadrics, i.e.

$$\text{Status} = \begin{cases} \text{In collision, } \|\boldsymbol{x}_0\| \leq \|\boldsymbol{x}_{mb}(\boldsymbol{\psi})\| \\ \\ \text{Separated, } \|\boldsymbol{x}_0\| > \|\boldsymbol{x}_{mb}(\boldsymbol{\psi})\| \end{cases} \tag{6.1}$$

The key computational step is to search for the point $\boldsymbol{x}_{mb}$ based on the parametric surface and the point $\boldsymbol{x}_0$. Concretely, $\boldsymbol{x}_{mb}$ is to be found such that its distance to the line $l_{O\boldsymbol{x}_0}$ is zero. This subproblem is addressed in both 2D and 3D cases as follows.

6.2.2 2D Case

Let θ parameterize the closed-form Minkowski sum boundary curve in $\mathbb{R}^2$, then the elements of the points $\boldsymbol{x}_{mb}$ and $\boldsymbol{x}_0$ are defined as $\boldsymbol{x}_{mb}(\theta) = [x_{mb}(\theta), y_{mb}(\theta)]^\top$ and $\boldsymbol{x}_0 = [x_0, y_0]^\top$ respectively. The distance between $\boldsymbol{x}_{mb}$ and $l_{O\boldsymbol{x}_0}$ can be

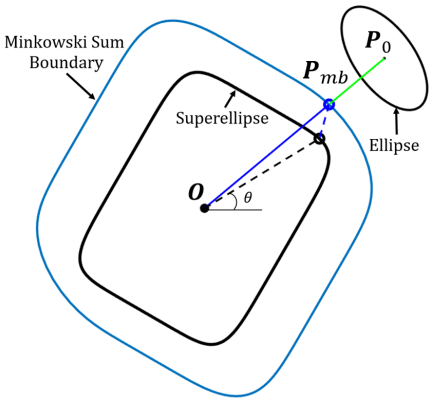


Figure 6.2: A demonstration of the collision detection scheme in 2D. $\boldsymbol{P}_{mb}$ is parameterized by θ, which can be obtained by solving for Eq. (6.3). In this situation, the ellipse is separated from the superellipse since $\|\boldsymbol{P}_0\| > \|\boldsymbol{P}_{mb}\|$.

written as

$$d(\boldsymbol{x}_{mb}(\theta), l_{O\boldsymbol{x}_0}) = \frac{|y_0 x_{mb} - x_0 y_{mb}|}{\sqrt{x_0^2 + y_0^2}} \quad (\|\boldsymbol{x}_0\| \neq 0), \tag{6.2}$$

where $l_{O\boldsymbol{x}_0}$ denotes the line connecting the origin and the center of ellipse.

Setting the distance to be zero gives the objective function

$$F(\theta) = y_0 x_{mb} - x_0 y_{mb} = 0. \tag{6.3}$$

Solving for θ gives the parameter that defines the point $\boldsymbol{x}_{mb}(\theta)$ on the Minkowski sum boundary. Moreover, if $\theta \in [0, \pi)$, the solution is unique, and the distance expression is valid as long as the point $\boldsymbol{x}_0$ does not coincide with the origin. Figure 6.2 demonstrates the collision detection scheme in 2D.

6.2.3 3D Case

Two parameters (i.e. η and ω) are used to define the explicit expression of the Minkowski sum boundary surface. Therefore, the distance between

$\boldsymbol{x}_{mb}(\eta,\omega) = [x_{mb}(\eta,\omega), y_{mb}(\eta,\omega), z_{mb}(\eta,\omega)]^\top$ and the line defined by $\boldsymbol{x}_0 = [x_0, y_0, z_0]^\top$ and the origin can be expressed as

$$d(\boldsymbol{x}_{mb}(\eta,\omega), l_{O\boldsymbol{x}_0}) = \frac{\|\boldsymbol{x}_{mb} \times \boldsymbol{x}_0\|}{\|\boldsymbol{x}_0\|} \quad (\|\boldsymbol{x}_0\| \neq 0). \tag{6.4}$$

Setting the distance to be zero gives the objective function

$$\boldsymbol{F}(\eta,\omega) = \boldsymbol{x}_{mb}(\eta,\omega) \times \boldsymbol{x}_0 = \begin{bmatrix} y_{mb}z_0 - z_{mb}y_0 \\ z_{mb}x_0 - x_{mb}z_0 \\ x_{mb}y_0 - y_{mb}x_0 \end{bmatrix} = \boldsymbol{0}. \tag{6.5}$$

Note that this is not an over-constrained system of equations since each equation in $\boldsymbol{F}$ can be derived from the other two equations. Hence, in the 3D case, there are two equations and two unknowns, and the solution is unique up to a reflection with respect to the origin when $\eta \in [0, \pi/2)$ and $\omega \in [0, \pi)$.

6.2.4 The Collision Detection Algorithm

Based on the derivations of separation checking between a point and the parametric closed-form Minkowski sum boundary, an algorithm is proposed for collision detection between an ellipsoid and a general convex differentiable surface. In this work, superquadric model is chosen as an example for such a convex surface.

Algorithm 1 solves the collision detection problem in general. In practice, the most computational intensive step is finding the root of the nonlinear equation $\boldsymbol{F}(\boldsymbol{\psi}) = \boldsymbol{0}$ in Step 4. Since there is no simple closed-form solution for this equation, numerical root finding needs to be done.

Algorithm 1: Collision Checking Procedure for Ellipsoidal and Superquadric objects

Input: S_a (semi-axes lengths $\boldsymbol{a}$, epsilons $\boldsymbol{\epsilon}_a$, orientation R_a, position of center $\boldsymbol{t}_a$);
E_b (semi-axes lengths $\boldsymbol{b}$, orientation R_b, position of center $\boldsymbol{t}_b$).
Output: Status (0 for separated, 1 for in collision).

1 Construct the point on the Minkowski Sum boundary $\boldsymbol{x}_{mb}(\boldsymbol{\psi})$ via Eq. (3.50) ;
2 Solve the objective function $\boldsymbol{F}(\boldsymbol{\psi}) = \boldsymbol{0}$ for $\widetilde{\boldsymbol{\psi}}$ via Eq. (6.3) or Eq. (6.5) ;
3 Compute the point $\boldsymbol{x}_{mb}(\widetilde{\boldsymbol{\psi}})$;
4 Compare the magnitudes $\|\boldsymbol{x}_{mb}(\widetilde{\boldsymbol{\psi}})\|$ and $\|\boldsymbol{x}_0\|$, and determine the Status via Eq. (6.1).

6.3 Proximity Queries Solutions

Apart from collision (contact) detection, more interesting aspects for the relative positions between two objects from the closed-form Minkowski sum expressions can be derived, such as the distance (or penetration depth if in collision) between two objects, closest point on each object surface (or the touching point is they kiss each other). Therefore, the collision detection algorithm is extended to compute these values related to proximity computations.

At first, the distance between two bodies A, B can be defined as

$$D(A, B) = \min \|\boldsymbol{x}_a - \boldsymbol{x}_b\| \ (\forall \boldsymbol{x}_a \in A, \boldsymbol{x}_b \in B). \tag{6.6}$$

Alternatively $D(A, B) = D(0, A \ominus B)$, where $A \ominus B$ is the Minkowski difference between the two bodies. Therefore, the distance between the two bodies can be computed as the distance between the origin and the Minkowski difference between them, and this is the core concept of the well-known distance algorithm GJK [55].

This concept is also closely related to the Minkowski sums in the field of robot motion planning in constructing the configuration space obstacles (C-obstacle). If one point $\boldsymbol{x}_{b0}$ on the body B is tracked and the Minkowski sums with A is computed, then B will be shrunk into the reference point $\boldsymbol{x}_{b0}$ and the boundary of A will be expanded. Mathematically, the Minkowski sum $A \oplus B \doteq A \ominus B(0)$, where $B(0)$ is the body B when centered at the origin. Then the distance $D(A, B)$ is also equivalent to the distance between $\boldsymbol{x}_{b0}$ and $A \oplus B$, i.e. $D(A, B) = D(\boldsymbol{x}_{b0}, A \oplus B)$. The proof can be seen from the geometric point of view.

Since the expression for the closed-form Minkowski sum between an ellipsoid (i.e. E_b) and any convex differentiable surface (i.e. S_a) (Eq. (3.51)) is obtained, computing the distance and the closest points can be formulated as an optimization problem as follows.

6.3.1 Distance Computations

The distance between S_a and E_b can then be computed as

$$D(S_a, E_b) = \min \|\boldsymbol{x}_{mb}(\boldsymbol{\psi}) - \boldsymbol{x}_0\|. \tag{6.7}$$

The minimizer $\widehat{\boldsymbol{\psi}}$ parameterizes the point on the Minkowski sum boundary that is closest to the center of the ellipsoid $\boldsymbol{x}_0$, and the magnitude of their difference is the distance of the two bodies. If the point is inside, then the resulting distance is the penetration depth of E_b inside S_a.

6.3.2 Closest Points and Associated Normal Vectors

The closest point on each body satisfies that their distance is $D(S_a, E_b)$. Observe that the second term of Eq. (3.51), i.e. $\boldsymbol{x}_n(\widehat{\boldsymbol{\psi}})$, satisfies the implicit expression of $E_b(0)$, i.e.,

$$\begin{aligned}\boldsymbol{x}_n^\top R_b \Lambda^{-2}(\boldsymbol{b}) R_b^\top \boldsymbol{x}_n &= \left[\frac{R_b \Lambda^2(\boldsymbol{b}) R_b^\top \nabla\Phi}{\|\Lambda(\boldsymbol{b}) R_b^\top \nabla\Phi\|}\right]^\top R_b \Lambda^{-2}(\boldsymbol{b}) R_b^\top \left[\frac{R_b \Lambda^2(\boldsymbol{b}) R_b^\top \nabla\Phi}{\|\Lambda(\boldsymbol{b}) R_b^\top \nabla\Phi\|}\right] \\ &= \frac{\nabla^\top\Phi R_b \Lambda^2(\boldsymbol{b}) R_b^\top \nabla\Phi}{\nabla^\top\Phi R_b \Lambda^2(\boldsymbol{b}) R_b^\top \nabla\Phi} = 1.\end{aligned} \tag{6.8}$$

Then, $\boldsymbol{x}_0 - \boldsymbol{x}_n(\widehat{\boldsymbol{\psi}})$ is a point on ∂E_b. The fact that $\boldsymbol{x}(\widehat{\boldsymbol{\psi}})$ is a point on ∂S_a gives

$$D(\boldsymbol{x}(\widehat{\boldsymbol{\psi}}), \boldsymbol{x}_0 - \boldsymbol{x}_n(\widehat{\boldsymbol{\psi}})) = D(\boldsymbol{x}_{mb}(\widehat{\boldsymbol{\psi}}), \boldsymbol{x}_0) = D(S_a, E_b). \tag{6.9}$$

Therefore, the closest point to the body E_b on body S_a is $\boldsymbol{x}(\widehat{\boldsymbol{\psi}})$ and the closest point to the body S_a on body E_b is $\boldsymbol{x}_0 - \boldsymbol{x}_n(\widehat{\boldsymbol{\psi}})$.

From the geometric point of view, the normal vector for the closest points (or the contact normal if two bodies touch) can be computed as

$$\boldsymbol{n}_1 = \frac{\boldsymbol{x}_0 - \boldsymbol{x}_{mb}(\widehat{\boldsymbol{\psi}})}{\|\boldsymbol{x}_0 - \boldsymbol{x}_{mb}(\widehat{\boldsymbol{\psi}})\|} = \frac{\nabla\Phi(\boldsymbol{x}(\widehat{\boldsymbol{\psi}}))}{\|\nabla\Phi(\boldsymbol{x}(\widehat{\boldsymbol{\psi}}))\|}. \tag{6.10}$$

This is because that the closest point of a surface to a point can also be defined such that the connecting line between them is on the direction of the surface normal. Also for two surfaces, the line that connecting the two closest points should be simultaneously perpendicular to both the surface tangent.

This derives another interesting property that the normal of the Minkowski sum surface at $\boldsymbol{x}_{mb}(\widehat{\boldsymbol{\psi}})$ is the same with the normal of the body surface ∂S_a at

$\boldsymbol{x}(\widehat{\boldsymbol{\psi}})$. Note that this property can be generalized to any such pair of points on the Minkowski sum surface and ∂S_a.

6.4 Accuracy Metric for Mesh-based Algorithms

The performance of any collision detection algorithm is evaluated by both efficiency and accuracy, the former of which can be compared by the running time. The accuracy is an equally important judgment for a good collision checking algorithm, but is more difficult to define. Here, an accuracy evaluation metric for the contact space is introduced. The metric is defined based on the volume of all configurations where the collision occurs. Such volume can be computed via the Principal Kinematic Formula [29] as

$$I(S_a, E_b) = \int_{\mathrm{SE(n)}} i(S_a \cap gE_b) dg, \tag{6.11}$$

where S_a and E_b are superquadrics and ellipsoids in $\mathbb{R}^n$ respectively, $g = (R, \boldsymbol{t}) \in \mathrm{SE}(n)$ describes the pose of E_b (i.e. $gE_b \doteq RE_b + \boldsymbol{t}$), dg is the natural bi-invariant integration measure for $\mathrm{SE}(n)$ [25] and $i(S_a \cap gE_b)$ is an indicator function defined as

$$i(S_a \cap gE_b) = \begin{cases} 1, & S_a \cap gE_b \neq \varnothing \\ \\ 0, & S_a \cap gE_b = \varnothing \end{cases} \tag{6.12}$$

Since inscribed meshes or bounding volumes make approximations to the actual objects, there is the possibility that the inexact algorithms return "no collision" when there is actually a collision, and vice versa. Therefore, computing the relative volume of all the possible collision configurations

gives a metric to evaluate the accuracy of performing collision detection. The relative volume can be computed as

$$\gamma = \frac{I(Mesh_a, Mesh_b)}{I(S_a, E_b)} \times 100\%, \tag{6.13}$$

where $Mesh_a$ and $Mesh_b$ are the two objects enclosed by meshes. Note that for an exact algorithm, such as our proposed method or the Algebraic Separation Condition (ASC) [152] for ellipsoidal models, $\gamma = 100\%$.

Explicitly, for the cases of $\mathrm{SE}(2)$, Eq. (6.11) can be calculated as [29]

$$I_{\mathrm{SE}(2)}(S_a, E_b) \; = \; 2\pi[A(S_a) + A(E_b)] + L(S_a)L(E_b), \tag{6.14}$$

where, $A(\cdot)$ and $L(\cdot)$ denote the area and perimeter of the objects respectively. And for $\mathrm{SE}(3)$, the corresponding simple expression for Eq.(6.11) is

$$\begin{aligned} I_{\mathrm{SE}(3)}(S_a, E_b) =& 8\pi^2[V(S_a) + V(E_b)] \\ &+ 2\pi F(\partial E_b)M(\partial S_a) + 2\pi F(\partial S_a)M(\partial E_b), \end{aligned} \tag{6.15}$$

where $V(\cdot)$ is the volume of a body in $\mathbb{R}^n$, $F(\cdot)$ and $M(\cdot)$ are the surface area and the integral of mean curvature of the bounding surface enclosing a spatial body, respectively.

6.5 Benchmark with Existing Methods

In this section, the computational time and accuracy of the proposed collision detection algorithm is benchmarked with some existing algorithms. For the ellipsoid-ellipsoid case, both ASC and GJK methods are compared; while for the ellipsoid-superquadrics case, only GJK is compared. All the algorithms

Table 6.1: A list of benchmarks for the Algorithm 1

Dimension	Ellipsoid-Ellipsoid	Ellipsoid-Superquadrics
2D	ASC, GJK (E), GJK (Mesh)	GJK (Mesh)
3D	ASC, GJK (E), GJK (Mesh)	GJK (Mesh)

are implemented in C++, and the benchmarks run in an Intel Core i7 CPU at 3.60GHz.

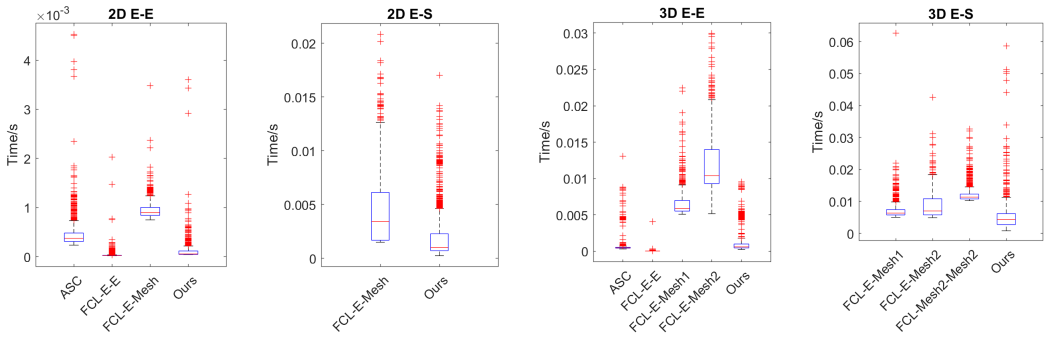
6.5.1 Benchmark Parameters and Notations

To make a fair comparison, the same parameters are input into different algorithms. The parameters include the geometry (i.e. semi-axes lengths and exponents) and configuration of the objects (i.e. orientation and location). For ASC and the proposed Minkowski-based algorithms, those parameters are directly used in the algorithms. For GJK, from the Flexible Collision Library (FCL), they need to be converted to specific representation objects. For the shape representation, the built-in "Ellipsoid (E)" and "Mesh" objects for ellipsoids and "Mesh" for superquadrics (S) are used. Table 6.1 lists the algorithms and object shapes being used for benchmarks.

The effects of using different mesh densities are also compared by varying the number of vertices and facets that constructs the convex bodies of the objects. To generate those meshes, a popular computational geometry library in C++, i.e. CGAL [48], is applied. Detailed vertices and surface information for the generated meshes are provided in Table 6.2 with the notation of each mesh used in the rest of the content.

Table 6.2: Parameters for ellipsoid and superquadrics meshes.

Dimension/Object	Notation	# Vertices	# Facets
2D/Ellipse (E)	Mesh	50	48
2D/Superellipse (S)	Mesh	50	48
3D/Ellipsoid (E)	Mesh1	25	268
3D/Superquadrics (S)	Mesh1	25	260
3D/Ellipsoid (E)	Mesh2	100	540
3D/Superquadrics (S)	Mesh2	100	534



(a) 2D, E-E **(b)** 2D, E-S **(c)** 3D, E-E **(d)** 3D, E-S

Figure 6.3: Running time comparisons with existing methods and different object representations. For ASC and our proposed method, the shape and configuration parameters are directly used; For GJK, different object representations provided in FCL are compared, which are labeled as "FCL-Object1-Object2". In the labels,"E" stands for "ellipse/ellipsoid" and "S" stands for "superellipse/superquadrics".

6.5.2 Running Time Results

The benchmarks are conducted by fixing one body and randomly generating 1000 poses of the other. Then the running time of collision checking for each configuration is recorded. Figure 6.3 shows the running time comparisons, with the line segment being standard deviation and its center being the mean.

Table 6.3: Collision detection accuracy for each case in 2D and 3D.

Case	Algorithm/Objects pair	Accuracy (γ)
2D/E-E	ASC	100%
2D/E-E	GJK/E-E	79.53%
2D/E-E	GJK/E-Mesh	82.25%
2D/E-E	Ours	100%
2D/E-S	GJK/E-Mesh	88.65%
2D/E-S	Ours	100%
3D/E-E	ASC	100%
3D/E-E	GJK/E-E	38.18%
3D/E-E	GJK/E-Mesh1	42.08%
3D/E-E	GJK/E-Mesh2	60.98%
3D/E-E	Ours	100%
3D/E-S	GJK/E-Mesh1	28.62%
3D/E-S	GJK/E-Mesh2	72.33%
3D/E-S	GJK/Mesh1-Mesh2	74.77%
3D/E-S	Ours	100%

6.5.3 Accuracy Evaluation of Collision Detection

Based on the metric described in Sec. 6.4, the accuracy γ is compared for different object representations in both 2D and 3D. Table 6.3 shows the comparisons of the collision detection accuracy for each object representation pair for the corresponding experimental trial in both 2D and 3D.

6.6 Discussion

With the current implementation, the Minkowski-based collision checker is competitive with some of the existing state-of-the-art algorithms. The significant advantages of the proposed algorithm are the direct use of the shape and configuration parameters without the need of meshes or bounding volumes, and the ability to extend to more complex shapes embedded in the

Euclidean space.

In the Ellipsoid-Ellipsoid (E-E) case, both Algebraic Separation Condition and Minkowski-based methods provide exact collision detection, with the difference being the former only solves a cubic or quartic polynomial, which can be efficient. The Minkowski-based method, however, requires solving for the roots of a nonlinear equation. It is essential to apply an efficient nonlinear optimization algorithm and solver (the trust-region algorithm is chosen, and the nonlinear root finding method from the well-known "Eigen" library in C++ is used). The results show that, in terms of running time, ours is competitive with ASC, and even outperforms in the 2D case.

Another remarkable advantage of the proposed method is that it can be extended to more complex shapes. Although this chapter only derives concrete expressions and conducts experiments for superquadrics, the closed-form Minkowski sum can be applied to *any* convex and differentiable surface embedded in $\mathbb{R}^n$. Consequently, the proposed general algorithm can deal with the collision detection problem between an ellipsoid and any object shape as long as it has implicit and parametric expressions.

For objects with complex shapes, generating meshes or bounding with volumes are common methods, with GJK being one of the most rigorous and efficient collision checkers for those primitives. The proposed algorithm is compared with GJK in both the E-E and E-S collision checking scenarios using different mesh densities. It turns out that by using the "Ellipsoid" object in FCL, GJK runs remarkably faster, the reason of which being the bounding volumes have fewer vertices. For the benchmarks, the "Ellipsoid"

object is used for one agent and generate different meshes by varying the number of vertices and facets for the other agent. Also, meshes for both objects are generated to show the changes in the performance of the GJK detection algorithm with respect to different mesh densities. It is obvious that as the number of vertices and facets increase, GJK takes longer time to execute. This gives limitations to the GJK algorithm, which is an inexact checker and depends significantly on the quality of the meshes or bounding volumes. Our algorithm, on the other hand, is an exact checker, with the accuracy $\gamma = 100\%$, therefore outperforms GJK when the number of vertices becomes larger.

Moreover, as the accuracy comparisons show, the volume of all the possible collision configurations computed from the mesh is always smaller than the one of the exact representation of the object, since the former always gives a lower bound for the object if the vertices are generated on the boundary. As a result, even when the two objects collide, inexact algorithms might sometimes return false negative results, and the probability of returning the true results is reflected by the accuracy.

6.7 Chapter Summary

This chapter investigates the collision and proximity query problems between an ellipsoid and a superquadric object. The geometric formulations are based on the closed-form Minkowski sums — a parametric expression that enlarges the superquadrics surface boundary and shrinks the ellipsoid into a point in $\mathbb{R}^n$. A collision detection algorithm is proposed, which involves computing the closed-form Minkowski sum boundary and finding the relative position

between a point and a parametric surface. Based on the parametric expression of the closed-form Minkowski sum boundary, the shortest distance and nearest point on the surface with respect to an arbitrary given point can also be computed. With the same spirit of solving for a nonlinear optimization problem, the proximity queries between two objects can be answered. Furthermore, to evaluate the probability of returning the true collision results, a collision detection accuracy based on the relative volume of all collision configurations of the objects is introduced.

The performance of the proposed Minkowski-based algorithm is compared with some existing popular collision detection algorithms: Algebraic Separation Conditions (ASC) for the ellipsoid-ellipsoid case, and Gilbert-Johnson-Keerthi (GJK) for both the ellipsoid-ellipsoid and ellipsoid-superquadric cases. The majority of the computational time for the proposed method is spent on solving the roots of a nonlinear equation, and a numerical solver is used in practice. The GJK method depends on the complexity of the meshes belonging to the objects, and different numbers of vertices and facets are used and compared. The benchmark results show that the proposed method is competitive with ASC and GJK, and outperforms as the mesh density increases.

Chapter 7

The Kinematics of Containment for N-dimensional Ellipsoids

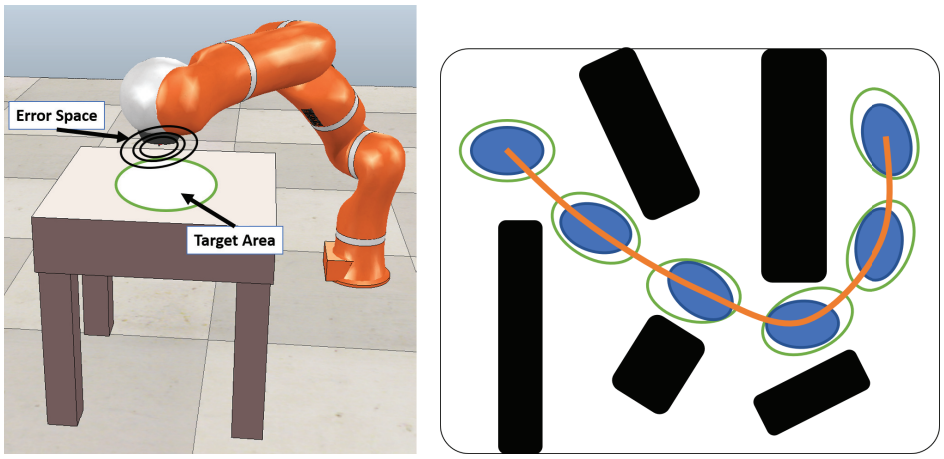
This chapter discusses the kinematics of containment problem for ellipsoids in $\mathbb{R}^n$. The kinematics of containment specifically describes the motion space of a rigid body being fully inside of a slightly larger one. Both algebraic and geometric aspects are studied in 2D and 3D cases, and the applications to some real robotic scenarios are discussed.

7.1 Introduction

Determining the allowable motions of an object in a structured environment is of interest in the field of robot motion planning [84], computer-aided design (CAD) [75], and automated assembly [62]. This problem can be interpreted as detecting whether an object in a specific pose (a position and orientation pair) is fully contained inside of a void, and computing how much volume such poses occupy in the whole configuration space (C-space).

Concretely in an assembly task, for instance, a robot manipulator is picking an object and trying to assemble it into another part. Due to the errors propagated from each joint, even with a fixed input control signal, the pose of the end effector always has uncertainties. As a result, the union of the object at all possible ending poses formulates an *error space*. The target area can be inscribed by ellipsoids, because of the simplicity of the shape that uses fewer parameters. Once the error space is fully contained inside the target area, the object can always be safely placed into the target. Further, the description of all the allowable motions in C-space of the object to be successfully placed into the target evaluates the robustness of the manipulator. Developing convenient methods for such problems would be helpful for the interval analysis of kinematic errors. And an important real-life application is the manipulators design for automated micro-assembly tasks, which requires precise robotic tools with guaranteed performance metric [115].

Furthermore, for a robot motion planning problem, as another example, sample-based planners such as PRM [77], RRT [86], RRT-Connect [82] have been well known for years and proved to be probabilistic complete and efficient in practice. However, large amounts of computations for collision detection between the robot and obstacles are required when dealing with narrow passage problems. Therefore, it is beneficial to develop efficient configuration construction and connection strategies in the C-space so that the traditional collision checking can be avoided. For a robot whose parts are rigidly connected to each other, it is possible to encapsulate it by a slightly larger convex shape, denoted as a *void*. Then the robot can be able to move



(a) Parts-handling for a robot manipulator with errors on the end effector.

(b) Path planning for an elliptical mobile robot being enclosed by a larger elliptical void.

Figure 7.1: Demonstration of the examples as motivations to the Kinematics of Containment theory.

safely inside the void. Once a *convex* subspace of the space of all its allowable motions can be computed, any path inside that subspace is guaranteed to be collision-free. Then, connecting two safe configurations remains simply finding a path within the convex subspace. Figure 7.1 demonstrates the assembly process and the motion planning problem described above, which motivates the work in this chapter.

One way to query whether the moving ellipsoid at a specific configuration is in collision with the fixed one is to perform collision detection, where [36], [73] give fast ways to check collisions. Moreover, [69] computes the signed distance between two overlapping ellipsoids, which gives an algorithm to check whether one ellipsoid is contained in another. However, collision detection cannot fully describe the C-space of the allowable motions of the moving object, and this is where the concept of the Kinematics of Containment

(KC) [35] fits in.

The KC theory discusses how to identify and describe the allowable motions of a convex body being fully contained inside a slightly larger one, and provides an efficient way to compute the range of the restricted motions. The KC theory is further applied in [103], where a closed-form hyper-spherical representation as a lower bound for the allowable motions when the convex objects are ellipsoids was proposed. The contributions are summarized as follows:

- Two lower bounds of the allowable motions based on the algebraic and geometric conditions of containment are developed respectively;
- Efficient containment checking process of a specific configuration for each lower bound is proposed;
- The computations of the occupied volumes for the two proposed lower bounds are performed and compared;

The remainder of this chapter is organized as follows. Section 7.2 formulates the algebraic and geometric conditions for one n-dimensional ellipsoid being fully contained in another. Further in Sec. 7.3, a convex lower bound is computed for the allowable motions of the smaller ellipsoid based on the approximated algebraic condition of containment. In Sec. 7.4, a geometric lower bound is calculated based on the closed-form Minkowski difference between two ellipsoids. To make the lower bounds useful, in Sec. 7.5, the containment checking processes are introduced for the two lower bounds and the computations of the occupied volumes of allowable motions in C-space.

In Sec. 7.6 and 7.7, numerical simulations are conducted in the 2D and 3D cases respectively, and the performance of the two proposed lower bounds are compared. In Sec. 7.8, applications are discussed on a configuration connection strategy in a robot motion planning paradigm, and the error estimations and evaluations of a parts-handling task for a robot manipulator. Section 7.9 summarizes the whole chapter.

7.2 The Algebraic and Geometric Conditions of Containment

7.2.1 Configuration Space of an N-Dimensional Ellipsoid

The allowable motions of the moving ellipsoid E_a can be described by the displacement of its center (as a translation $\boldsymbol{t} \in \mathbb{R}^n$) and the orientation (as a rotation $R \in \mathrm{SO}(n)$) with respect to the fixed ellipsoid E_b. Such a rotation and translation pair forms a Lie Group called "Pose Change Group" (PCG) as $(R, \boldsymbol{t}) \in \mathrm{PCG}(n) \doteq \mathrm{SO}(n) \times \mathbb{R}^n$ [31]. The corresponding Lie algebra can be obtained by logarithm map as $\boldsymbol{\xi} = [\boldsymbol{\omega}^\top, \boldsymbol{t}^\top]^\top \in \mathbb{R}^{n(n+1)/2}$. The Lie Algebra element can be transformed back to the Lie Group by exponential map, i.e. $(\exp(\widehat{\boldsymbol{\omega}}), \boldsymbol{t}) \in \mathrm{PCG}(n)$ [1].

An element in $\mathrm{PCG}(n)$ specifies a configuration of the moving ellipsoid, and all of the configurations formulate the configuration space (C-space) [98]. The subset of the whole C-space where the moving ellipsoid is fully contained in another fixed ellipsoid without any collision is denoted as the "Kinematics

[1] The mappings for the pose change group between the Lie Group and its Lie Algebra are different than the exponential and logarithm for $\mathrm{SE}(n)$.

of Containment C-space (KC C-space)". The choice of PCG(n) provides a correct and natural way to represent a pose of the body and compute the change of poses as seen from the fixed world reference frame. In particular, the vector $\boldsymbol{t}$, which is the actual translation as seen in the world frame, remains the same when performing the exponential mappings between Lie group and Lie algebra. This is convenient from the computational aspect. On the other hand, for the conventional SE(n) representation, the translation part will change and have different meanings through the exponential maps.

Moreover, the volume computations of the KC C-space in PCG(n) follows the same fashion as in SE(n), where the rotation and translation parts can be split as

$$\int_{\mathrm{PCG}(n)} f(R,\boldsymbol{t})d\boldsymbol{t}dR = \int_{\mathrm{SO}(n)} \int_{\mathbb{R}^n} f(R,\boldsymbol{t})d\boldsymbol{t}dR, \tag{7.1}$$

where $f(\cdot)$ denotes a general function.

7.2.2 The Algebraic Condition of Containment

The semi-axes of E_a and E_b are denoted as $\boldsymbol{a} = [a_1, a_2, ..., a_n]^\top, \boldsymbol{b} = [b_1, b_2, ..., b_n]^\top \in \mathbb{R}^n$ respectively. By substituting the explicit expression of the moving ellipsoid E_a into the implicit expression of the fixed ellipsoid E_b that is aligned with the world frame, the algebraic condition for E_a to move inside E_b without collision can be written as [103]

$$(R_a\Lambda(\mathbf{a})\boldsymbol{u} + \boldsymbol{t}_a)^\top \Lambda^{-2}(\mathbf{b})(R_a\Lambda(\mathbf{a})\boldsymbol{u} + \boldsymbol{t}_a) \leq 1, \tag{7.2}$$

where $\boldsymbol{u}$ is the explicit expression of an n-dimensional sphere with $\|\boldsymbol{u}\| = 1$.

For this highly nonlinear expression, a small angle approximation can

make it much simpler, where some better properties, such as convexity, can be proved. If the rotation of E_a is restricted, the rotation part calculated by exponential map can be approximated to the first order as

$$R_a = \exp(\widehat{\omega}_a) \approx \mathbb{I} + \widehat{\omega}_a, \tag{7.3}$$

where $\mathbb{I} \in \mathbb{R}^{n \times n}$ denotes an identity matrix, $\boldsymbol{\omega} \in \mathfrak{so}(n)$ is the Lie algebra of R.

Substituting Eq. (7.3) into Eq. (7.2) and grouping parameters ($\boldsymbol{u}$) and variables ($\boldsymbol{\omega}$ and $\boldsymbol{t}$) gives the approximation of the left-hand side of the algebraic condition of containment as

$$C_{\boldsymbol{u}}(\boldsymbol{\xi}) \doteq \boldsymbol{\xi}^\top H(\boldsymbol{u})\boldsymbol{\xi} + \boldsymbol{h}^\top(\boldsymbol{u})\boldsymbol{\xi} + c(\boldsymbol{u}), \tag{7.4}$$

where $H(\boldsymbol{u}) \in \mathbb{R}^{n(n+1)/2 \times n(n+1)/2}$, $\boldsymbol{h}(\boldsymbol{u}) \in \mathbb{R}^{n(n+1)/2}$ and $c(\boldsymbol{u}) \in \mathbb{R}$. The first order algebraic condition of containment is then defined as

$$C_{\boldsymbol{u}}(\boldsymbol{\xi}) \leq 1. \tag{7.5}$$

The approximation is a subset (or a lower bound) of the actual algebraic condition of containment in the sense that Eq. (7.5) implies Eq. (7.2) and the reverse is not true. This statement is verified by numerically sampling 10000 random configurations and testing the status of Eq. (7.2) and Eq. (7.5) respectively. The result is shown as a confusion matrix in Tab. 7.1 for both the 2D and 3D cases. From the experiment, when the approximation returns "True" (E_a is fully contained in E_b), the actual containment condition is always "True", which implies that all the configurations that satisfy the approximation also satisfy the actual algebraic condition of containment.

Table 7.1: Confusion matrix for the actual algebraic condition in Eq. (7.2) of containment and its approximation in Eq. (7.5).

Actual (2D)	Approx. (2D)		Actual (3D)	Approx. (3D)	
	True	False		True	False
True	1858	1771	True	2404	951
False	0	6371	False	0	6645

7.2.3 The Geometric Condition of Containment

The KC C-space boundary can also be determined in a geometric way: For each fixed orientation of E_a, the trajectory of its center when just touching E_b is generated by the Minkowski difference between the two ellipsoids. And the whole actual KC C-space boundary can be constructed as a union of those Minkowski differences at all possible orientations.

For two n-dimensional ellipsoids, the Minkowski difference can be calculated in explicit closed-form efficiently [164], by first shrinking E_a into a sphere (E'_a) and computing an offset curve. The constraint for such derivation requires the curvature of the ellipsoid at every point after shrinking must be smaller than the curvature of the sphere. Another implementation for Minkowski difference between two ellipsoids is introduced in Ellipsoidal Toolbox [83], which is used throughout this paper to generate the exact KC C-space as a reference for comparison.

The explicit boundary of the Minkowski difference cannot be applied directly in the KC theory, because it is nontrivial to determine whether a point is inside only from the knowledge of the parametric boundary expression. As a result, it is important to find a lower bound for the KC C-space that has a simple expression, making such a querying process easy and fast. Inspired by

the shrinking process in the closed-form solutions, a geometric lower bound can be obtained from computing the extreme distance for the sphere to move along each semi-axis of the ellipsoid in the shrunk space.

7.3 A Convex Lower Bound Based on the Approximated Algebraic Condition of Containment

This section starts from showing that the approximation of the algebraic condition is convex, so that given some configurations, the interior of their convex hull is a safe subset and can be treated as a lower bound of the KC C-space. Then this convex lower bound can be constructed as a convex polyhedron using several extreme vertices.

7.3.1 Convexity of the Approximated Algebraic Condition of Containment

The approximated algebraic condition must be satisfied over all $\boldsymbol{u}_i$ in the unit sphere, which is equivalent to

$$\max_{\forall \boldsymbol{u}_i} C_i(\xi) = \max_{\forall \boldsymbol{u}_i} [\boldsymbol{\xi}^\top H(\boldsymbol{u}_i)\boldsymbol{\xi} + \boldsymbol{h}(\boldsymbol{u}_i)^\top \boldsymbol{\xi} + c(\boldsymbol{u}_i)] \leq 1 \tag{7.6}$$

Firstly the convexity of the left-hand side of (7.6) is proved as follows.

Proof. For any fixed $\boldsymbol{u}_i$ ($i = 1, ..., m$), given $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2 \in \mathbb{R}^{n(n+1)/2}$,

$$\begin{aligned}
&C_i(\alpha\boldsymbol{\xi}_1 + (1-\alpha)\boldsymbol{\xi}_2) - [\alpha C_i(\boldsymbol{\xi}_1) + (1-\alpha)C_i(\boldsymbol{\xi}_2)] \\
=&[(\alpha\boldsymbol{\xi}_1 + (1-\alpha)\boldsymbol{\xi}_2)^\top H(\boldsymbol{u}_i)(\alpha\boldsymbol{\xi}_1 + (1-\alpha)\boldsymbol{\xi}_2) \\
&+ \boldsymbol{h}(\boldsymbol{u}_i)^\top(\alpha\boldsymbol{\xi}_1 + (1-\alpha)\boldsymbol{\xi}_2) + c(\boldsymbol{u}_i)] \\
&-[\alpha(\boldsymbol{\xi}_1^\top H(\boldsymbol{u}_i)\boldsymbol{\xi}_1 + \boldsymbol{h}(\boldsymbol{u}_i)^\top\boldsymbol{\xi}_1 + c(\boldsymbol{u}_i)) \\
&+ (1-\alpha)(\boldsymbol{\xi}_2^\top H(\boldsymbol{u}_i)\boldsymbol{\xi}_2 + \boldsymbol{h}(\boldsymbol{u}_i)^\top\boldsymbol{\xi}_2 + c(\boldsymbol{u}_i))] \\
=&-\alpha(1-\alpha)[(\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)^\top H(\boldsymbol{u}_i)(\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)], \forall\alpha \in [0,1].
\end{aligned} \tag{7.7}$$

The above expression is non-positive if and only if $(\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)^\top H(\boldsymbol{u}_i)(\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2) \geq 0$, or equivalently, $H(\boldsymbol{u}_i)$ is symmetric positive semi-definite. The direct calculations for the quadratic part gives that $\forall\boldsymbol{\xi} \in \mathbb{R}^{n(n+1)/2}$,

$$\boldsymbol{\xi}^\top H(\boldsymbol{u}_i)\boldsymbol{\xi} = (\widehat{\omega}_a\Lambda(\boldsymbol{a})\boldsymbol{u}_i + \boldsymbol{t}_a)^\top\Lambda^{-2}(\boldsymbol{b})(\widehat{\omega}_a\Lambda(\boldsymbol{a})\boldsymbol{u}_i + \boldsymbol{t}_a). \tag{7.8}$$

Since $\Lambda^{-2}(\boldsymbol{b})$ is diagonal with non-negative entries on diagonal, Eq. (7.8) ≥ 0, which means that $H(\boldsymbol{u}_i)$ is symmetric and positive semi-definite. Hence, each condition function $C_i(\boldsymbol{\xi})$ is convex. And since maximization preserves convexity [15], $\max C_i(\boldsymbol{\xi})$ is convex, which concludes the proof. □

From the convexity of $\max C_i(\boldsymbol{\xi}_j)$, if for two extreme points $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$,

$$\max C_i(\boldsymbol{\xi}_j) \leq 1, j = 1, 2$$

hold, then for points on the line segment between them, $\alpha\boldsymbol{\xi}_1 + (1-\alpha)\boldsymbol{\xi}_2, \forall\alpha \in$

$[0, 1]$,

$$\max C_i(\alpha\boldsymbol{\xi}_1 + (1-\alpha)\boldsymbol{\xi}_2) \leq \alpha \max C_i(\boldsymbol{\xi}_1) + (1-\alpha)\max C_i(\boldsymbol{\xi}_2) \leq 1$$

is also satisfied. Hence points inside the convex hull of the extreme points also satisfy the approximated algebraic condition of containment.

7.3.2 Finding extreme vertices that represent the polyhedron

Now the extreme points of the polyhedron are searched by the following 2 cases: (1) extreme points that lie on each axis of the C-space; (2) points that have the largest magnitude.

Extreme points in each axis can be simply found by fixing the other axis lengths to zero. Since in each axis, there are 2 extreme points (positive and negative), a total of $2n$ points can be obtained for the first case, where n is the dimension of the configuration space.

For the vertices that have largest magnitude, the squared norm of $\boldsymbol{\xi}$ is to be maximized, with the constraint being the algebraic condition as

$$\boldsymbol{\xi}^* = \arg\max \boldsymbol{\xi}^\top \boldsymbol{\xi} \text{ s.t. } C_i(\boldsymbol{\xi}) \leq 1 \ (i = 1, ..., m). \tag{7.9}$$

Since the objective function is quadratic, the solutions for each variable have 2 possibilities. Therefore, the total number of solutions can be up to 2^n. However, not all of those possibilities are feasible solutions, meaning that validations are required by substituting back to the constraint inequalities.

7.4 A Geometric Lower Bound Based on the Minkowski Difference between Two Ellipsoids

Here a convex polyhedron is constructed as a lower bound for the Minkowski difference boundary by computing the extreme points at each semi-axis of the ellipsoid in the shrunk space.

7.4.1 Extreme Distance that a Sphere Can Move Along Each Semi-axis of an Ellipsoid in $\mathbb{R}^n$

Inspired by the derivations of Minkowski difference, the geometric lower bound can be obtained from the extreme distance that E'_a can move along each semi-axis of E'_b in the shrunk space. It can be further observed E'_b is still an ellipsoid, whose semi-axis length and orientation can be computed by eigenvalue decomposition of the transformed shape matrix $T\Lambda^{-2}(\mathbf{b})T = R'\Lambda^{-2}(\boldsymbol{b}')R'^{\top}$. For simplicity, E'_b is further rotated by $R'^{\top}$ to align with the world frame. The extreme distance at each semi-axis happens when E'_a just touches E'_b, the condition of which is stated as follows.

Suppose $\boldsymbol{x}_0 = [x_1, x_2, ..., x_n]^{\top} \in \partial E'_b$ is a point on the ellipsoid surface, it should also be on the surface of the sphere, and the outward normals of both surfaces at $\boldsymbol{x}_0$ should lie on the same line. Then, the problem becomes

simultaneously solving

$$\boldsymbol{x}_0^\top \Lambda^{-2}(\boldsymbol{b}')\boldsymbol{x}_0 = 1, \tag{7.10a}$$

$$(\boldsymbol{x}_0 - d\boldsymbol{e}_i)^\top (\boldsymbol{x}_0 - d\boldsymbol{e}_i) = r^2, \tag{7.10b}$$

$$2\Lambda^{-2}(\boldsymbol{b}')\boldsymbol{x}_0 = 2k(\boldsymbol{x}_0 - d\boldsymbol{e}_i), \tag{7.10c}$$

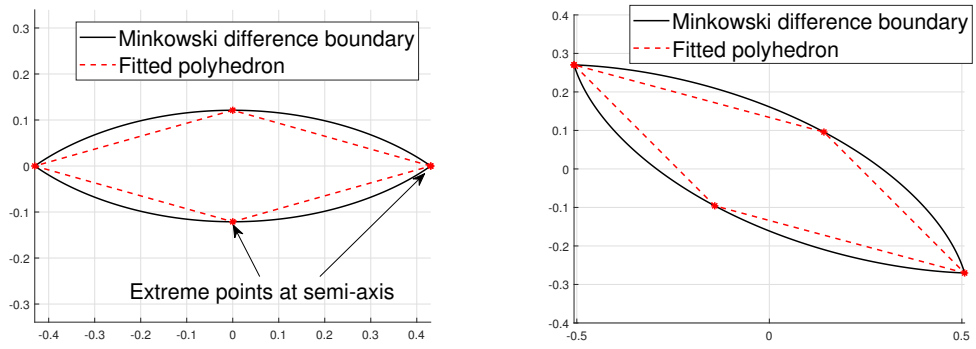
where d is the extreme distance at the i^{th} semi-axis, and k is a scalar indicating the outward normal of both surfaces lies on the same line. The solution can be obtained as

$$d_i^* = \begin{cases} \dfrac{1}{b'_{j^*}}\sqrt{\left(b'^2_{j^*} - b'^2_i\right)\left(r^2 - b'^2_{j^*}\right)}, & r \geq b'^2_{j^*}/b'_i \\ b'_i - r, & r < b'^2_{j^*}/b'_i, \end{cases} \tag{7.11}$$

where $j^* = \arg\max_{j \neq i}(b'_j)$. The proof of this result is available in Appendix A.4.

7.4.2 Polyhedron as a Lower Bound for the Minkowski Difference Boundary at Each Orientation of the Moving Ellipsoid

From the result (7.11), a convex polyhedron in the shrunk space can be constructed by the extreme points at each semi-axis of the ellipsoid. The polyhedron is guaranteed to be in the interior of the true Minkowski difference boundary since Minkowski difference between two convex sets are convex. Also, since affine transformation preserves the convexity [15], transforming back from the shrunk space still gives a convex polyhedron which is a



(a) Extreme points in the shrunk space.

(b) Transformed extreme points in the actual C-space.

Figure 7.2: Demonstration of polyhedron lower bound for Minkowski difference boundary in the shrunk space and actual C-space.

lower bound for the Minkowski difference between the two original ellipsoids. Figure 7.2 shows the idea of polyhedron lower bound for the Minkowski difference in the shrunk space and the actual C-space.

The polyhedron introduced above is defined at one specific orientation of the moving ellipsoid, and the union of the polyhedron subset at all orientations of E_a formulates the geometric lower bound of the KC C-space. Note that, this lower bound is no longer a convex polyhedron any more, but it is relatively simple in terms of querying an interior point and computing the volume.

7.5 Containment Checking and Volume Computations for the Lower Bounds of KC C-space

This section gives a review of how to query a point $P_{test} \in \mathbb{R}^n$ inside a polyhedron and compute the volume of such an n-dimensional polyhedron. The processes can be directly applied for the *Convex Lower Bound*, and extended

for the *Geometric Lower Bound*.

Suppose the convex polyhedron is constructed by m vertices $\{P_i\} \in \mathbb{R}^n, i = 1, ..., m$. It can be decomposed into a union of disjoint simplexes in $\mathbb{R}^n$ by Delaunay triangulation [43]. For each simplex with $n+1$ vertices, i.e. $P_i^s (i = 0, ..., n)$, the point inside should satisfy

$$P_{test} = \sum_{i=0}^{n} \lambda_i P_i^s, \text{ where } \lambda_i \in [0,1] \text{ and } \sum_{i=0}^{n} \lambda_i = 1. \tag{7.12}$$

This condition can be formed in matrix form as

$$\begin{bmatrix} P_{test} \\ 1 \end{bmatrix} = \begin{bmatrix} P_0^s & P_1^s & ... & P_n^s \\ 1 & 1 & ... & 1 \end{bmatrix} \begin{bmatrix} \lambda_0 & \lambda_1 & \dots & \lambda_n \end{bmatrix}^\top, \ \lambda_i \in [0,1] (\forall i) \tag{7.13}$$

The point P_{test} is inside the simplex if the solution of the matrix equation Eq. (7.13), $[\lambda_0, ..., \lambda_n]^\top$, satisfies $\lambda_i \in [0,1] (\forall i)$. Further, this point is inside the polyhedron if it is inside any decomposed simplex.

Given the vertices of a convex polyhedron, the volume can be computed as a sum of all the volumes of the decomposed simplexes as

$$\mathcal{V}_{poly} = \sum_{i=1}^{m} \mathcal{V}_{simplex}^{(i)} (P_0^s, P_1^s, ..., P_n^s), \tag{7.14}$$

where $\mathcal{V}_{simplex}^{(i)} (P_0^s, P_1^s, ..., P_n^s)$ denotes the volume of the i-th simplex with $n+1$ vertices $P_0^s, P_1^s, ..., P_n^s$. The volume of a simplex in $\mathbb{R}^n$ defined by the $n+1$ vertices can be calculated as [139]

$$\mathcal{V}_{simplex} = \left\| \frac{1}{n!} \det(P_1^s - P_0^s, P_2^s - P_0^s, ..., P_n^s - P_0^s) \right\| \tag{7.15}$$

For the Convex Lower Bound, the above computations can be applied directly; and further for the Geometric Lower Bound, since at each fixed

orientation, the KC C-space is a convex polyhedron, the same process can be used.

The containment checking for the Geometric Lower Bound is given as follows: once a configuration $\xi_{test} = [\omega_{test}^\top, \boldsymbol{t}_{test}^\top]^\top$ is given, the whole space is transformed via the knowledge of ω_{test}, and the translation part $\boldsymbol{t}_{test}$ is queried in the shrunk space. For the volume, since the polyhedron vertices are aligned with the semi-axes of the ellipsoid, and the two vertices on one semi-axis are symmetry about the origin, the volume of the polyhedron becomes

$$\mathcal{V}'_{geo}(R_a) = \frac{2^n}{n!} \prod_{i=1}^{n} d_i^*. \tag{7.16}$$

The actual volume after the inverse affine transformation can be computed as

$$\mathcal{V}_{geo}(R_a) = \det(TR'_a)\mathcal{V}'_{geo}(R_a) = \frac{2^n}{n!} \left|\Lambda(\boldsymbol{a}/r)\right| \prod_{i=1}^{n} d_i^*. \tag{7.17}$$

The volume computed at each orientation of E_a is a function of the rotation group, then, the total volume can be calculated by integration over all rotations as

$$V_{total} = \int_R \mathcal{V}_{geo}(R) dR. \tag{7.18}$$

Explicitly for the case $SO(2)$,

$$V_{total}^{SO(2)} = \int_{-\pi}^{\pi} \mathcal{V}_{geo}(\theta) d\theta; \tag{7.19}$$

and for the case $SO(3)$,

$$V_{total}^{SO(3)} = \int_{\|\boldsymbol{\omega}\|<\pi} \mathcal{V}_{geo}(R(\boldsymbol{\omega})) \left|\det(J(\boldsymbol{\omega}))\right| d\boldsymbol{\omega}, \tag{7.20}$$

where analytically $\left|\det(J(\boldsymbol{\omega}))\right| = \frac{2(1-\cos\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2}$ [25].

7.6 Numerical Simulation Studies in 2D

This section computes the convex and geometric lower bounds for the KC C-space in one rigid body 2D ellipse case. We define $\boldsymbol{a} = [a_1, a_2]^\top$ and $\boldsymbol{b} = [b_1, b_2]^\top = (1+\epsilon)\boldsymbol{a}$ as the semi-axis lengths of the moving and fixed ellipses E_a and E_b. The constrained motion of E_a within E_b is described as a rotation and translation pair $(R, \boldsymbol{t}) \in \mathrm{PCG}(2)$. For the convex lower bound which applies the algebraic conditions, the expressions of matrix H, vector $\boldsymbol{h}$ and scalar c have been calculated explicitly in [103]. And for visualization and comparison purposes, the shape of the true Minkowski difference boundary is constructed to illustrate the relationships between the actual KC C-space and the two proposed lower bounds. All the experiments are implemented in Matlab 2017a and run in an Intel Core i7-4790 CPU @ 3.60GHz.

7.6.1 Visualizations and Containment Checking Validations of the Two Lower Bounds

The Convex Lower Bound is construct by first finding extreme points at each C-space axis as follows. For the two translational axis (x and y), the extreme points are located at $x_{ex} = \pm\epsilon a_1$ and $y_{ex} = \pm\epsilon a_2$ respectively. For the rotational axis (θ), the extreme points can be found, in closed-form, as $\theta_{ex} = \arctan(\theta_y/\theta_x)$, where

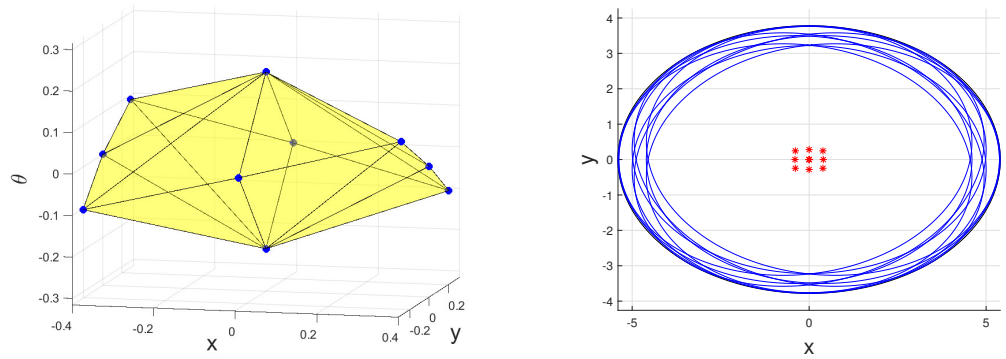
$$\theta_x = \sqrt{(1+\epsilon+\alpha)(1+\epsilon-\alpha)(1+\alpha(1+\epsilon))(1-\alpha(1+\epsilon))} \tag{7.21a}$$

$$\theta_y = \alpha\epsilon(2+\epsilon) \tag{7.21b}$$

Note that this solution can be found by equating the implicit expressions of the two ellipsoids. Then, for the points with largest magnitude, the convex constraint optimization with $\xi = [\theta, x, y]^\top$ is applied. Note that 8 results can be obtained since the cost function is quadratic, only 4 of them are valid by plugging back into the constraint functions. Thus, a total of 10 extreme points can be obtained to construct the polyhedron subspace from the configuration space.

Figure 7.3a demonstrates the proposed convex lower bound in C-space as a polyhedron made by 10 extreme vertices, which are plotted as big dots, and Fig. 7.3b plots the small ellipses in Euclidean space which are at the extreme configurations. To check the validity of the convex lower bounds, Fig. 7.4 shows the numerical results of the "point-in-polyhedron" test for 1000 randomly sampled configurations for E_a. Further, the ellipses whose configurations are inside the polyhedron are checked for collision with the larger ellipse using Eq. (7.2) numerically. The configuration points inside are indicated as plus signs while those in collision are marked as dots. The corresponding ellipses with those safe configurations are drawn in green whose center are marked as red asterisks. This visualization, along with the collision checking criteria, numerically verifies that the proposed convex lower bound gives the collision-free space and the querying procedure is numerically correct.

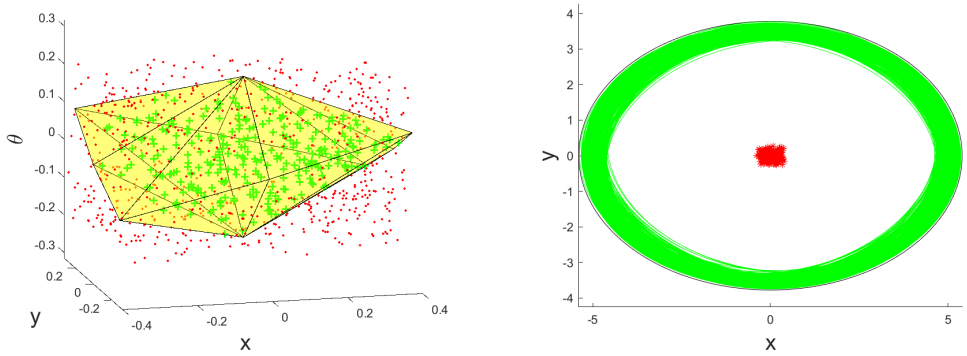
The same sampling and containment checking procedure are performed for the geometric lower bound. Figure 7.5a demonstrates the containment checking process with 1000 sampled configurations, and the numerical collision



(a) Convex lower bound by 10 extreme vertices. Blue dots: the extreme configurations; Shaded area: the convex polyhedral lower bound.

(b) Ellipses at the 10 extreme configurations. Blue ellipses: ellipses with the extreme configurations; Red asterisks: centers of the blue ellipses.

Figure 7.3: Visualizations for the Convex Lower Bound.

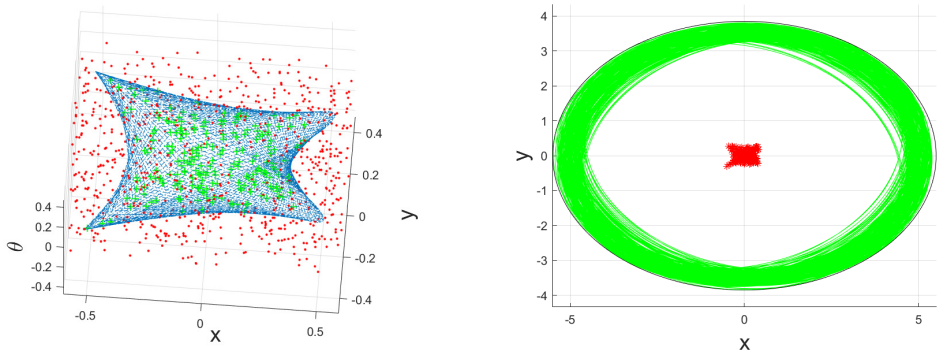


(a) Convex lower bound with sampled configurations. Dots: outside the polyhedron; plus signs: safe configurations inside the polyhedron.

(b) Ellipses with the safe configurations. Green ellipses: ellipses with safe configurations; Red asterisks: centers of the green ellipses.

Figure 7.4: Validation of the containment checking procedure for Convex Lower Bound.

checking are performed for each safe configuration for a double confirmation. The blue dashed surface visualizes the shape of the geometric lower bound, where the rotational angles are uniformly sampled between the maximum and minimum allowable angles. The number of the angles does not play



(a) Geometric lower bound with sampled configurations. Dots: outside the lower bound; plus signs: safe configurations inside the lower bound.

(b) Ellipses with the safe configurations inside the geometric lower bound. Green ellipses: ellipses with safe configurations; Red asterisks: centers of the green ellipses.

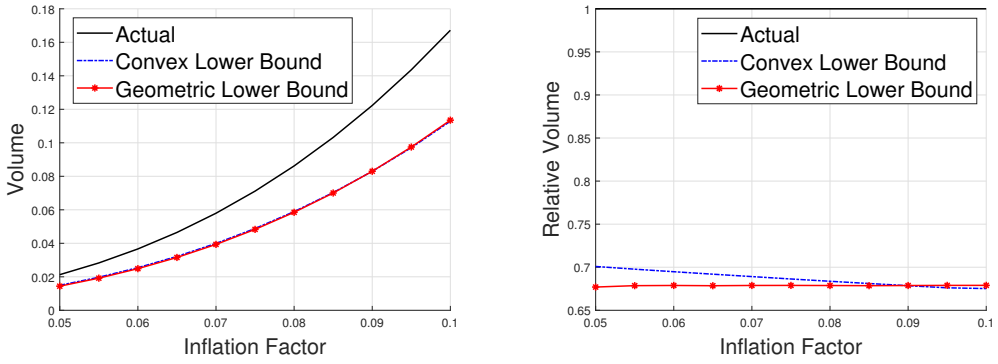
Figure 7.5: Validation of the containment checking procedure for Geometric Lower Bound.

an important role, since it does not affect the containment checking process. Plus signs and dots indicates the collision-free and in-collision configurations, respectively. Figure 7.5b shows the safe poses of the smaller ellipsoid in Euclidean space.

7.6.2 Volume Comparisons of Different Lower Bounds

The volumes of the KC C-space computed from both the algebraic and geometric methods are compared. The exact volume of KC C-space is calculated as a reference by numerically integrating the volume of the exact Minkowski difference between the two ellipses over $\mathrm{SO}(2)$, using Eq. (3.59).

The simulations are conducted by: (1) varying the inflation factor ϵ between E_a and E_b with the semi-axis lengths fixed as $\boldsymbol{a} = [5, 3.5]^\top$ and $\boldsymbol{b} = (1+\epsilon)\boldsymbol{a}$; and (2) varying the aspect ratio $\alpha = a_1/a_2$ for E_a with fixed longer



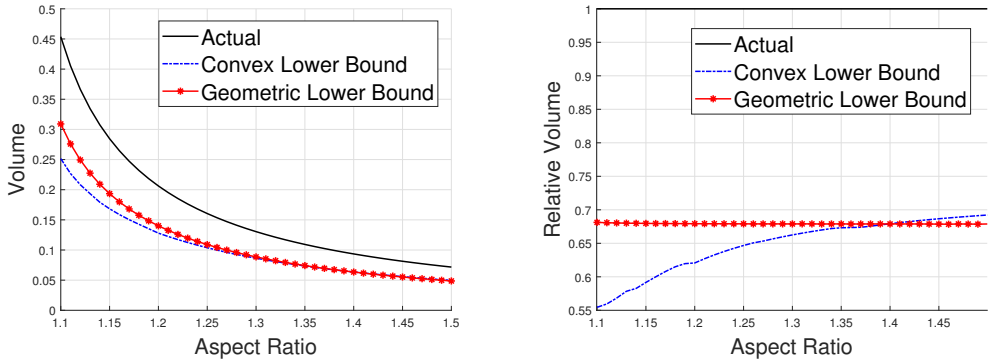
(a) Absolute volume. **(b)** Relative volume.

Figure 7.6: Comparisons in 2D for the volumes of different lower bounds of the KC C-space with different inflation factors.

semi-axis length at $a_1 = 5$ and inflation factor at $\epsilon = 0.08$.

Figure 7.6 shows the volume comparisons of the KC C-space for different methods, where Fig. 7.6a shows the absolute volume and Fig. 7.6b shows the relative volumes compared to the one generated by the Minkowski difference. The convex lower bound occupies a slightly larger space when the inflation factor is small, but becomes smaller as the inflation factor increases. This indicates that the small angle assumption and the first-order approximation works well when the rotation is restricted, but drops accuracy when there is more freedom for E_a to rotate. The geometric lower bound, on the other hand, performs more stable in terms of the relative volume. This shows that the polygon generated by extreme points at each semi-axis is a good choice to approximate the Minkowski difference boundary in the shrunk space.

Figure 7.7a compares the volume for different methods with the aspect ratio of the ellipse varying at the range $\alpha \in [1.1, 1.5]$, and Fig. 7.7b shows the relative volumes with the one generated by Minkowski difference. As



(a) Absolute volume. **(b)** Relative volume.

Figure 7.7: Comparisons in 2D for the volumes of different lower bounds of the KC C-space with different aspect ratios.

the aspect ratio increases, the volume of allowable motion decreases, but the relative volume for the convex lower bound increases. When the aspect ratio is close to 1, the ellipses are close to circles. Therefore, E_a has larger free space to rotate, which makes the first-order approximation less accurate. Then, the convex lower bound performs worse. But the geometric lower bound still works much more stable with the changes of the aspect ratio.

7.7 Numerical Simulation Studies in 3D

This section further conducts simulation studies for the 3D ellipsoid case, whose configuration space is now 6 dimensional, i.e $\boldsymbol{\xi} = [\omega_1, \omega_2, \omega_3, x, y, z]^\top \in \mathbb{R}^6$. Since it is not possible to visualize a 6D space, only the query process for sampled configurations is performed. For a double confirmation, collision detection are performed for each method. Then, the volume of each lower bound of KC C-space are computed for comparisons.

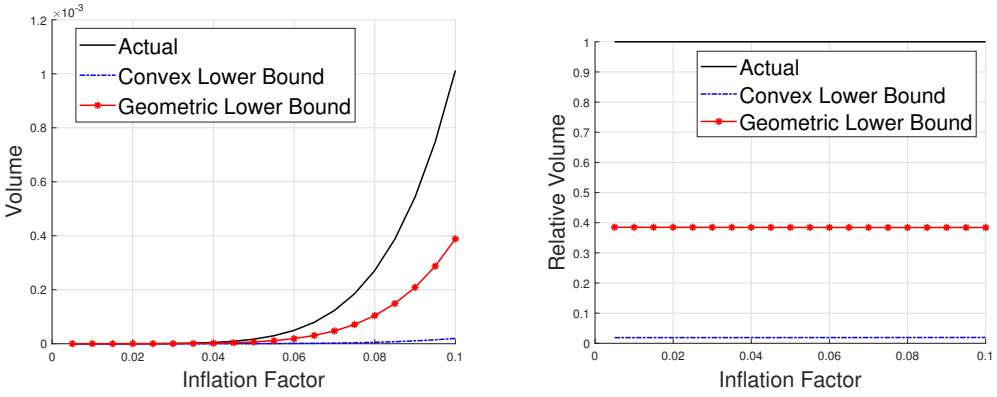
7.7.1 Containment Checking Validations of the Two Lower Bounds

The extreme points in translation axes are, similar to 2D, $x_{extreme} = \pm\epsilon a_1$, $y_{extreme} = \pm\epsilon a_2$ and $z_{extreme} = \pm\epsilon a_3$ respectively. To find the extreme points in rotational axes, the 3D space is projected onto the x, y-, x, z-, y, z- plane. Therefore, the problem shrinks to the 2D case of finding the extreme rotational points at each plane, then the number of extreme points is 12. The extreme points with largest magnitude can be obtained by the constraint convex optimization from Eq. (7.9), which gives 64 possible solutions and only 32 are valid. Combined with the extreme points at each axis, a total of 44 extreme points can be obtained to create the 6D polyhedron. For numerical validations, 1000 configurations are randomly sampled, and those points are queried in the 6D polyhedron. The results are compared with the collision detection based on the exact algebraic condition of containment to verify our theory.

For the geometric lower bound, the orientation of each of the 1000 sampled configurations is firstly computed, and the translation part is checked whether to be inside the polyhedron lower bound for the Minkowski difference boundary. Further validations of the exact algebraic condition of containment are performed for double confirmation.

7.7.2 Volume Comparisons of the Two Lower Bounds

The volumes of the convex and geometric lower bounds are compared here. The parameters of the comparisons are inflation factors and aspect ratios. For the inflation factors, the semi-axis lengths of E_a are set to be $\boldsymbol{a} = [4, 2.5, 2]^\top$,

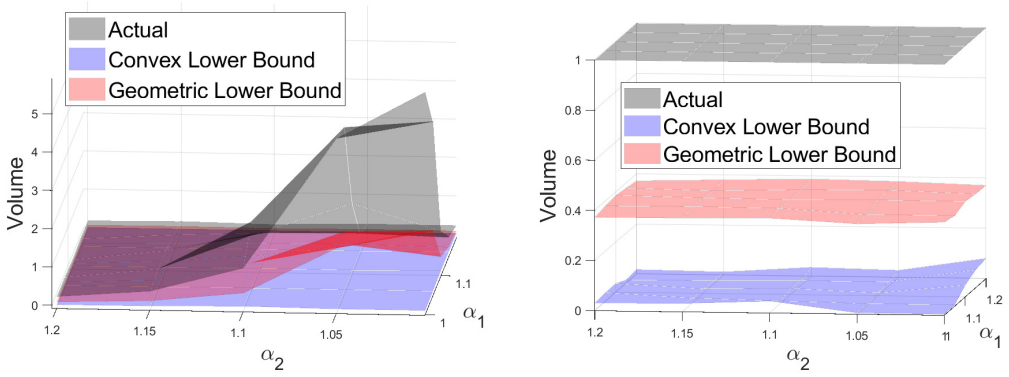


(a) Absolute volume. **(b)** Relative volume.

Figure 7.8: Comparisons in 3D for the volumes of different lower bounds of the KC C-space with different inflation factors.

and the inflation factors vary within the range of $\epsilon \in [0.01, 0.2]$. Figure 7.8 shows the volume comparisons between different lower bounds with respect to the inflation factor. The geometric lower bound occupies a much larger volume of the KC C-space than the convex lower bound. This means that when there are more degrees of freedom, the convex subset is no longer a good approximation of the entire KC C-space.

As a comparison over different aspect ratios, the largest semi-axis of E_a is fixed as $a_1 = 4$, and the other two semi-axis lengths are varied by different aspect ratios, i.e. $a_2 = a_1/\alpha_1, a_3 = a_1/\alpha_2$, where $\alpha_1, \alpha_2 \in [1, 1.2]$. Figure 7.9 shows the comparison results. As the aspect ratios increase, which gives more constraints for E_a to move, the volumes decreases, but the relative volumes are stable for both of the two lower bounds. Also, the geometric lower bound performs much better than the convex lower bound.



(a) Absolute volume. **(b)** Relative volume.

Figure 7.9: Comparisons in 3D for the volumes of different lower bounds of the KC C-space with different aspect ratios.

7.8 Applications

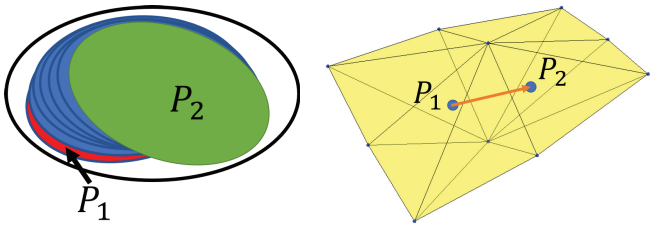
The kinematics of containment for ellipsoids has a wide range of real-life applications such as robot motion planning, parts-handling mechanisms, automated assembly, etc. For example, when planning a collision-free path for a mobile robot, the convex subspace of the allowable motions gives a solution for safe configuration connections; and when evaluating the robustness of a robot manipulator, the volume of the allowable motion space can be treated as a metric. This section discusses the potential applications of the proposed theory, and provides some simple examples that might lead to the future work.

7.8.1 Safe Configuration Connections for Robot Motion Planning Problems

Generating a collision-free path is an essential goal in robot motion planning problems. Existing effective algorithms include PRM, RRT and their variants.

The general spirit of these planners is to randomly generate samples in the configuration space and check for collisions with obstacles in the Euclidean space. The collision-free configuration is stored in either a graph or tree data structure. Then, to connect these valid configurations via edges, intermediate steps between the two nearest neighboring configurations are computed through interpolations. Despite the fact of simplicity and effectiveness in practice, such edge connection strategies are discrete, which significantly depends on the resolution of interpolations. Therefore, it might not provide a safe guarantee for the edges, especially in the case of narrow passage. This is where the propose "Convex Lower Bound" of the KC C-space fits in.

As illustrated in the introduction, consider a 2D path planning problem for a robot enclosed by an ellipse, and suppose the enclosing ellipse is bounded by a larger void. Then the actual robot can move slightly inside the larger elliptical void. Once the vertex is generated using the larger ellipse, all the allowable motions of the robot inside the void are guaranteed to be collision-free, and a KC C-space can be computed accordingly. Here, the *Convex Lower Bound* plays a necessary role to connect adjacent configurations, since any straight line segment within the convex subspace is guaranteed to be safe. If the two adjacent configurations are both inside the convex lower bound, then connecting them remains simply to find a straight line segment between them. This process connects a collision-free edge between the two configurations without interpolations as well as traditional collision checking calculations for the intermediate configurations between them. Figure 7.10 demonstrates the idea of configuration connections within the convex lower bound of the KC



(a) The robot motions in the Euclidean space.

(b) Vertex connections in the configuration space.

Figure 7.10: A demonstration of the configuration connection strategy. The robot is moving from P_1 to P_2 while staying fully contained in the larger ellipse (as shown in a). If the KC C-space is convex (as shown in b), then the path is guaranteed to be collision-free.

C-space.

The KC theories and the vertex connection strategy described above have been applied and shown a success in 2D motion planning problems in [126], where the robot is encapsulated by an ellipse. As an extension of the motion planner, it is important to tackle the more difficult 3D problems with the similar ideas. This chapter deals with the general n-dimensional case (particularly verified when $n = 3$), and will provide a more useful tool for the motion planning algorithm in 3D.

7.8.2 Error Analysis for Robot Manipulators

The *pick-and-place* task is a famous problem for robot manipulators, where the accuracy of handling parts requires the control of errors from the joints. Because of the unavoidable errors propagated from the joints, the end effector of the manipulator always has uncertainties. Suppose that the object to be handled is enclosed by a 3D ellipsoid, with its center of mass being a reference point. Then a body-fixed reference frame can be attached to that point, which

describes the configuration of the object, and all the possible configurations form a space of uncertain poses, denoted as *error space*. In practice, the error space can be constructed numerically by encapsulating an ellipsoid to the object at some sample ending poses. The target placing location can be inscribed by another ellipsoid that is slightly larger than the object, in order to give some clearance to put the object.

Therefore, for such a pick-and-place task, it is always important to:
(I) determine whether the error space is fully contained in the target area; and
(II) assess the robustness of the robot that can deal with error on its joints.

These two goals are closely related to the proposed theory, where the *Geometric Lower Bound* can be applied since it occupies larger volume in the C-space. The following example gives a numerical demonstration of how the two goals are addressed by using the theory of KC C-space, which is on a KUKA LWR robot shown in Fig. 7.1.

The simulation is conducted by first setting a target pose, and solving for the corresponding configuration in the joint space [40]. The object is predefined with fixed semi-axes lengths, and the target area is slightly larger with a fixed inflation factor ϵ. To model the uncertainty, a zero mean Gaussian white noise is added to each joint angle. The simulation is repeated with different standard deviation of the noise, and at each trial, 100 random poses of the object are placed accordingly. Table 7.2 shows the numerical settings of this example.

Problem (I) can be addressed by directly querying whether the error space

Table 7.2: Numerical settings of the error analysis for a pick-and-place task by KUKA LWR robot.

Description	Numerical Data
Ellipsoidal object	$\mathbf{a}_0 = [0.2, 0.15, 0.1]^\top$
Desired pose	$(R\|\mathbf{t}) = \left(\begin{array}{ccc\|c} 1 & 0 & 0 & 0.5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.15 \end{array}\right)$
Desired joint angles	$\mathbf{q} = [-0.7768, 0.1991, -0.1991, 1.6981, -1.6241, 1.9656, -0.9147]^\top$

is inside the target area. The simulation results demonstrate the correspondence between the inflation factor and the uncertainty of each joint with different noise levels. The inflation factor for the target area is determined numerically as the minimum number that the object is placed safely inside the target at all the possible poses. This gives a guidance for the design of the control method to limit the error within an acceptable range.

To assess how much error the robot manipulator can deal with, as stated in problem (II), the concept of *parts entropy* [130] is used here as an evaluation metric. The parts entropy evolving in time t for one object is originally defined as

$$S_h(t) = -\int_G h(g;t) \log h(g;t) dg. \tag{7.22}$$

For this application where the target is fixed and the object is moving, the distribution of the constrained motions of the object can be computed as $h(g) = 1/V$, where V is the volume of the allowable motion in PCG(3) [28]. The resulting parts entropy is therefore given by $S_h = \log V$. Since the volume is associated with the inflation factor, for each experimental trial, the parts entropy is computed. Figure 7.11 plots the relationships between the joint

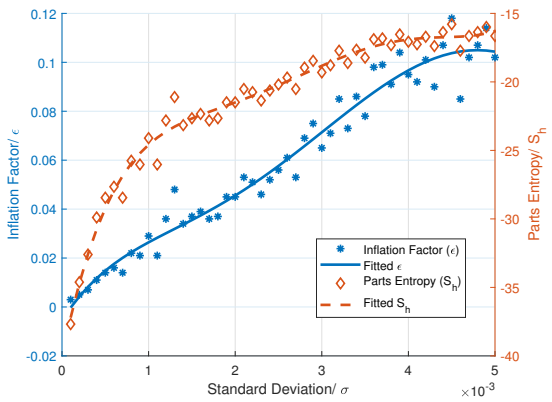


Figure 7.11: Simulation result for the pick-and-place task.

errors, inflation factors and the corresponding parts entropy.

The trend of the data points gives the relationships between the joint errors and robustness of the manipulator. As the error of each joint grows, the space to be placed needs to be larger to accommodate the noise, and the parts entropy becomes larger also. In the real application, on one hand, once the control parameters of the manipulator are well-tuned, one can refer to this figure to determine how large the target area is; while on the other hand, if the target space is chosen in advance, one can also read from the figure and find the requirement for the errors of the joints, which guides the control strategy.

7.9 Chapter Summary

The fields of automated assembly and robot motion planning deal with many problems of determining whether one object is contained in another, and how much space the smaller object can move without any collision with the larger one. This chapter applies the concept of the Kinematics of Containment and

investigates a special case when the arena is slightly larger than the moving object, both of which are ellipsoids. The algebraic condition of containment is reviewed and the geometric condition of containment is then introduced. Based on these two conditions, two lower bounds for the allowable motion in the configuration space are proposed, denoted as *Convex Lower Bound* and *Geometric Lower Bound* respectively. Containment checking process for a specific configuration and volume of motions within the lower bounds are introduced. To verify the theory, implementations and volume comparisons in 2D ellipses and 3D ellipsoids cases are performed. The results show that the Geometric Lower Bound occupies larger volume in the C-space than the Convex Lower Bound when the smaller ellipsoid has more freedom to move, and its relative volume to the actual KC C-space performs more stable with the change of the ellipsoid shapes. Finally, applications on a configuration connection strategy for path planning problems and a pick-and-place task with uncertainties on the end effector of a manipulator are studied and numerically demonstrated.

Chapter 8

Planning Feasible Paths for Complex Bodies in Cluttered Environment

This chapter proposes several novel efficient path planning frameworks for complex rigid and articulated bodies in cluttered environment. Geometrically feasible paths for robots in the environment including *narrow passages* are generated. The first algorithm constructs a graph structure for rigid bodies, which is able to answer multiple path searching queries. To further scale to higher dimensional problems, the second algorithm is hybrid with sampling-based planners by providing collision-free configurations a priori. The performances of the proposed algorithms are compared with the sampling-based planners on different kinds of environment. Physical experiments are also conducted using a humanoid robot.

8.1 Introduction

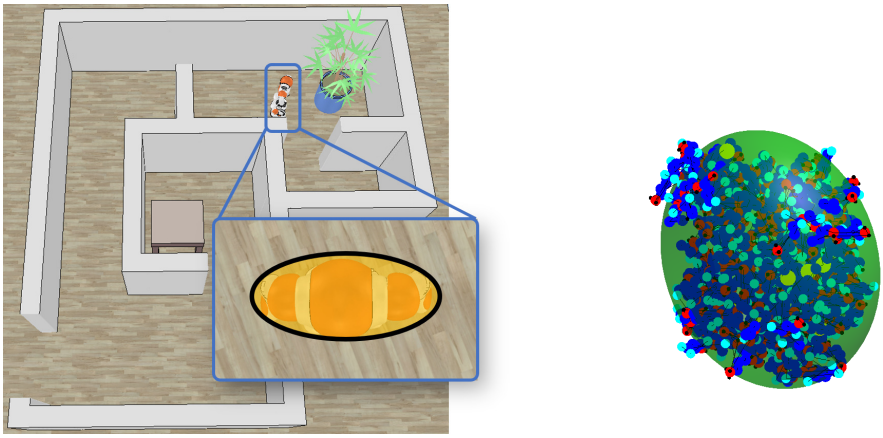
Sampling-based planners such as PRMs [77] and RRTs [86] (and a multitude of their extensions, e.g [18], [82]) have demonstrated remarkable success. The frameworks usually generate state samples randomly and perform explicit collision detection to assess their feasibility. These methods have had a profound impact both within robotics and across other fields such as molecular docking, urban planning, and assembly automation.

It is well known that despite the great success of these methods, the *narrow passage* problem remains a significant challenge. The reason is that, generally speaking, sampling-based approaches use a strategy of sampling states in the configuration space, followed by collision checking. When a robot and an obstacle are found to be in collision, the corresponding sample is discarded. Then, valid state configurations are connected by edges, where each edge is sub-sampled and collision checking is done along the edge. If any of the states on the edge corresponds to a collision, the whole edge is discarded. This approach works extremely well when the obstacles in the environment are sparse. But when there is a narrow passage, an inordinate amount of computational time is spent on the samples and edges that eventually will be discarded. To increase the probability of sampling valid configurations in a narrow passage, various methods have been proposed such as [66], [123], [133], [147]. However, there is still no guarantee of finding valid vertices efficiently within the corridor due to the probabilistic nature of sampling and collision checking. The first goal of this chapter is to:

1. Extend the previous algorithm of parameterizing the free space for single-body ellipsoidal robots avoiding ellipsoidal obstacles [165], so as to avoid traditional collision checking computations.

In the planner proposed here, the robot is enclosed by an union of ellipsoids, while the environmental features are represented as superquadrics. Ellipsoids have a wide range of applications in enclosing robots, which can provide geometrically tighter bounds and more physical meanings for rigid objects. For example, from the top view of a humanoid robot, the contour can be tightly encapsulated by an ellipse since its shoulder part on the side is narrower than the head [16], as shown in Fig. 8.1a. Also in the computational crystallography field, it is natural to enclose a protein by an ellipsoid, which is defined by the center of mass and moment of inertia from the atoms elements [134], as in Fig. 8.1b. This ellipsoidal model not only simplifies the geometrically complex representation, but also keep the physical information of the protein. Moreover, superquadrics generalize the characterization of ellipsoids by adding freedoms in choosing the power of the exponents rather than restricting to quadratics, which allows them to represent a wider range of the complex shapes while still requiring only a few parameters [2], [12], [71].

It is well known that for a rigid body with a fixed orientation in n-dimensional space, a *slice* of the configuration space (C-space) obstacle corresponding to this orientation is the Minkowski sum of the rigid body and the obstacles in the workspace [59], [76], [84], [113], denoted as a *C-layer* [92]. There is substantial literature on the computational complexity of Minkowski sums of polyhedra and faceted approximations of ellipsoids [13], [51], [60], [64],



(a) The projection contour from the top of a NAO humanoid robot is enclosed by an ellipse.

(b) The protein 1AKY is enclosed by an ellipsoid computed from the center of mass and moment of inertia of the atoms elements.

Figure 8.1: Examples of encapsulating robot by an ellipsoid.

[83]. Recently an exact closed-form formula for n-dimensional ellipsoids was introduced and discussed [164]. As a generalization of that, the closed-form Minkwoski sums of an ellipsoid and an arbitrary convex differentiable shape embedded in n-dimensional Euclidean space is presented here, with superquadrics being a typical example. This is another essential reason for the choice of superquadrics objects as environmental features in our new planner.

Minkowski sums characterize the C-space obstacles for the individual rigid components in a multi-body robot, and the feasibility of a robot's configuration corresponds to each rigid component of the robot in the complement of the union of C-space obstacles. Consequently collision-free samples can be generated. However, if one seeks to *connect* such samples using current sampling-based planning paradigms like PRM or RRT, then collision checking is still required. Therefore, the second goal of this chapter is to:

2. *Develop guaranteed safe and efficient methods for connecting configurations*

between different C-layers without performing traditional collision checking.

Two local planners are proposed here to efficiently generate valid paths connecting two configurations with different rotational components. A *local C-space* characterizes the restricted motions of the rigid parts of the robot to move within a small area, thereby allowing the robot to transfer via a guaranteed free area between two configurations. In addition, the idea of a *middle C-layer* provides possibilities for the robot to rotation without limitations. The minimum volume concentric ellipsoid (MVCE) is applied here, which allows the robot to transit between different C-layers via pure rotations. In addition, the middle C-layer allows the MVCE to translate in the free space, thereby allowing the vertices with different translational components to be connected. An extension is also proposed to compute a sweep volume by sliding the MVCE along the reference curve of the ellipsoidal part.

Despite the effectiveness of the proposed planning algorithm above, it only works for rigid bodies. For a typical robot with a mobile base and n revolute joints, the configuration space is $\mathrm{SE}(3) \times (S^1)^n$, where $(S^1)^n$ is an n-dimensional torus. Once the joint angles are fixed, one "shape" is defined and the robot can be considered as a rigid body. With the Highway RoadMap planner, the problem for this specific shape can be solved. However, when the joint angles are changing, the number of possible combinations is exponential with respect to the number of joints, therefore a *curse of dimensionality* will make the general planning problem intractable. Therefore, another important part of this chapter is to:

3. Propose solutions to deal with the high dimensionality burdens for an articulated body.

Sampling-based algorithms are scalable to high dimensional problems. They explore a discrete subset of the whole C-space by performing random sampling with explicit collision detection. The random sampling strategy guarantees probabilistic completeness, but can be expensive for dense environments where obstacles take more space. Therefore, this part of the chapter combines the advantages of sampling-based planners and combinatorial algorithms. The idea is to firstly compute valid configuration seeds by parameterizing the free space of different frozen configurations and feed these into sampling-based algorithms to solve for a path, resulting in a hybrid algorithm.

The key contributions of this chapter are summarized as follows:

- Construct a subgraph in each C-layer for complex rigid bodies;
- Propose two novel methods (i.e. local C-space and middle C-layer) to connect vertices on the roadmap between different C-layers;
- Propose a novel hybrid algorithm to generate collision-free configurations prior to the sampling-based planners so as to deal with high dimensional problems.

The rest of this chapter is organized as follows. Section 8.2 introduces the proposed Highway RoadMap algorithm for robots constructed by ellipsoidal parts. In Sec. 8.3, two novel methods are proposed to allow rotations in transferring between two configurations. Further in Sec. 8.4, a hybrid

algorithm is presented to deal with high dimensional problems. The efficiency and effectiveness of the proposed algorithms are evaluated by comparing with the most popular sampling-based planners from the well-known Open Motion Planning Library (OMPL) in Sec. 8.5 and Sec. 8.6. To further show the applicability of generating collision-free path efficiently in real-world problems, physical experiments are conducted to solve a walking path planning problem for a humanoid robot in cluttered environment in Sec. 8.7. Section 8.8 discusses the results. Section 8.9 concludes this chapter.

8.2 The Highway RoadMap Path Planning Algorithm for Robots with Ellipsoidal Components

The Highway RoadMap system is built based on the idea of parameterizing the free C-space at each orientation of the robot. At each sampled fixed orientation, a subset of the C-space that only contains translational motions is built, denoted as a *C-layer*. Then, to detect the collision-free space at each C-layer, a *sweep line* process is applied.

In particular, at each C-layer, The closed-form Minkowski sum and difference are computed between the robot and the obstacles and arenas, respectively [1]. Once the Minkowski operations are applied, the configuration space obstacles (C-obstacles) are generated. Then by sweeping a line throughout the C-space with a certain resolution, the free portion (C-free) can be detected and represented as line segments. A subset of the roadmap is then constructed by computing the middle point of the collision-free line segment as a vertex and

[1] Here the word "arena" denotes the bounded area in which the robot and obstacles are contained.

Algorithm 2: Highway RoadMap Algorithm

Input: *robot*; *obstacle*; *arena*
Parameter: N_{layer}: number of C-Layers; N_{line}: number of sweep lines
Output: *graph*; *path*

```
orientation ← SampleOrientations(N_layer);
foreach orientation do
    C_obstacle, C_arena ← MinkowskiOperations(robot, obstacle, arena);
    C_free ← SweepLineProcess(C_obstacle, C_arena, N_line);
    graph.Append( NewVertexAndEdge(C_free) );
end
graph ← ConnectAdjacentCLayers(graph);
path ← GraphSearch(graph)
```

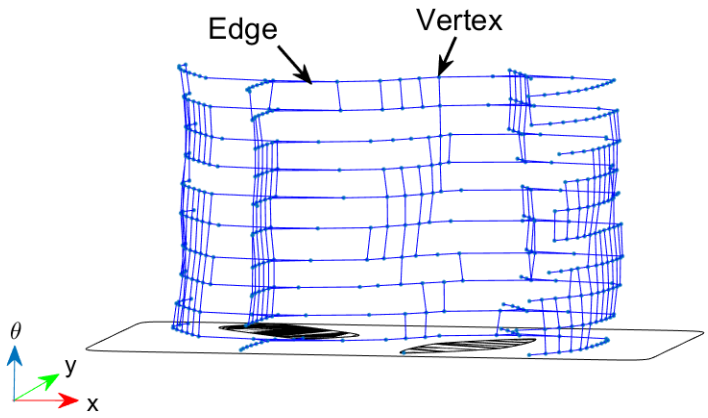


Figure 8.2: The fully connected graph structure, generated from one simulation trial. The vertical axis represents the rotational angle; dots are valid vertices and line segments are collision-free edges.

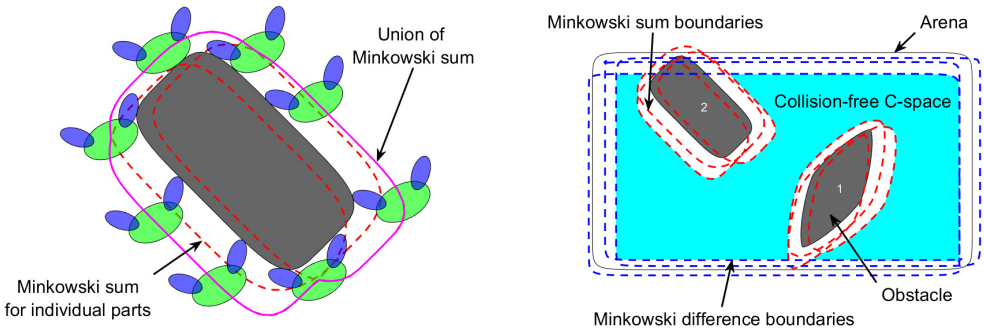
connecting edges between two vertices. The entire roadmap system can then be constructed by connecting vertices among adjacent C-layers. The general idea of constructing this graph-based roadmap system is illustrated in Algorithm 2, and a fully connected roadmap obtained by running our algorithm is shown in Figure 8.2 in the case of planar robot and obstacles to demonstrate the concept. The following subsections introduce and discuss the algorithm in detail.

8.2.1 Minkowski sums for a poly-ellipsoidal robot

Suppose that the robot is enclosed by a finite union of M ellipsoids $E_1, E_2, \ldots, E_M$, and the transformations between the base ellipsoid E_1 and other ellipsoids $E_2, E_3, \ldots, E_M$ are defined as $g_i = (R_i, \mathbf{t}_i)$ $(i = 2, \ldots, M)$ respectively. This definition of transformations is necessary for the Minkowski sums computations, and can be computed from the kinematic chains for multi-link robots. For a single ellipsoidal part, E_i, Minkowski sums can be computed with respect to its center via Eq. (3.51). Since such a boundary can be referred to any point, by offsetting $-\mathbf{t}_i$, the reference point of the Minkowski sum boundary will be at the origin of the base ellipsoid E_1. By using the property from Eq. (3.44), the resulting Minkowski sum of the poly-ellipsoidal robot is the union of the Minkowski sum of individual parts with respect to the same point at base ellipsoid center.

For computational purposes, the boundary surfaces of the robot's ellipsoidal parts and superquadric obstacles are discretized as meshes, whose vertices are defined by the angular parameters of superquadrics. And the Minkowski sum boundary is then characterized directly from the closed-form expression, the vertices of which are one-to-one mappings with that from superquadrics. Therefore, the complexity of our proposed closed-form characterizations only depends on the complexity of the superquadric surface.

The concatenation operation can also be applied in characterizing the free space. Let the collision-free C-space for each ellipsoid E_i be denoted as C_i $(i = 1, \ldots, M)$. Then the collision-free space for the whole robot can be characterized as the intersection of them as viewed in the body frame of base



(a) C-obstacle as the Minkowski sum boundaries of individual ellipsoidal bodies and their union.

(b) Collision free C-space as an intersection of free space for individual robot parts.

Figure 8.3: Algorithm for obtaining the characterization of the Minkowski sum between a convex superquadric and a union of ellipsoids.

ellipsoid, i.e. $C = C_1 \cap (g_2 \cdot C_2) \cap (g_3 \cdot C_3) \cap \cdots \cap (g_M \cdot C_M)$. This expression inspires an efficient routine for finding vertices in free space and constructing a roadmap system within one C-layer. Figure 8.3 shows the Minkowski sums of the poly-ellipsoidal robot at a fixed orientation and the collision-free C-space at one C-layer.

8.2.2 A sweep-line process for collision-free configuration space characterizations within one C-layer

Once C-obstacles are computed, the robot is now shrunk into a point at the center of its base ellipsoid. Then one natural way to characterize the free space is to generate a roadmap as a graph structure, with vertices being robot poses and edges indicating connectivity. The sweep-line process that generates this collision-free space characterization is reviewed here [38], [165].

At first, a set of parallel sweep lines are generated. Each sweep line

intersects all the C-space obstacles and arena boundary, with the intersecting points saved as pairs which formulates line segment intervals. Denoting the line segments within the obstacles as P_{O_i}, and those within the arenas as P_{A_i}, then the collision-free line segment P_{CF} for each sweep line can be represented as

$$P_{CF} = \bigcap_{i=1}^{M_A \times M} P_{A_i} - \bigcup_{j=1}^{M_O \times M} P_{O_i} \tag{8.1}$$

where M_A and M_O are numbers of superquadrics that represent arenas and obstacles respectively [162]. To make the distance from the vertex to obstacles as far as possible, the translational component of each vertex is computed as the middle point of each collision-free line segment. And this is a reason for the name of this proposed framework, which tries to construct a highway system with large safety margins to the obstacles. Figure 8.4 shows the decomposed C-space at one layer with collision-free cells highlighted by horizontal raster lines. Note that the resolution of these sweep lines are pre-defined, and the more dense of them, the more space can be identified. For a 3D map, raster lines are set to be parallel to z-axis, and the resolutions from both x and y directions are defined a priori.

8.3 Local planners for Vertex Connections between Adjacent C-layers

Since each C-layer only represents one orientation of the robot, one must connect the subgraphs among different C-layers so that the robot can transform between different layers by rotations. Two local planners are developed here that captures the connectivity of the rotational motions between adjacent

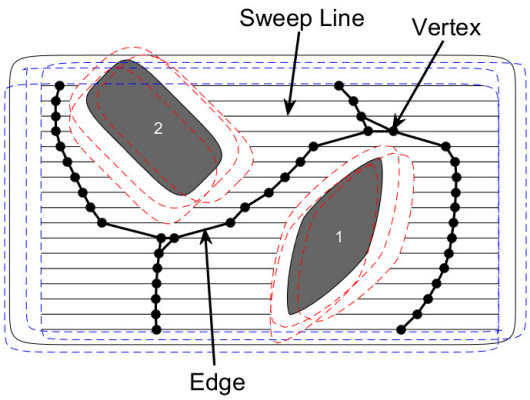


Figure 8.4: The sweep line process for detecting free space and construct sub-graph in one C-layer.

C-layers.

8.3.1 Local planner 1: the Local C-space

The first local planner constructs a continuous collision-free space that can enclose all the steps along the edge between two vertices. The core idea is to build a continuous convex C-space that allows the robot to move between two configurations in different C-layers.

Firstly, the ellipsoid that encapsulates a robot part is further enclosed by a slightly larger ellipsoid, i.e. scale the semi-axes by a factor of $\epsilon = 0.1$. Then the robot is allowed to move small amounts inside the larger ellipsoid without collisions. Such motions can be described locally in the C-space, denoted as *local C-space*. The local C-space becomes collision-free if the Minkowski operations are performed using the larger ellipsoid, and the descriptions of the local C-space can be done before building the roadmap a priori. Once the local C-space of the two vertices intersect, a new vertex can be generated within the intersecting area and connected to the two vertices. The following

subsections introduce in detail the characterizations of the local C-space and the procedure to connect two vertices by a collision-free path.

8.3.1.1 Characterization of the Local C-space

The concept of the kinematics of containment for n-dimensional ellipsoids is applied here [124], which characterizes the allowable motions of the smaller ellipsoid inside a larger ellipsoid. In particular, the *convex lower bound* is used here, which encloses several valid configurations by a convex hull and gives a convex polyhedron in the C-space of the smaller ellipsoid. This convex polyhedron describes a collision-free subspace and any path inside is guaranteed to be collision-free. The convex polyhedron is a lower bound of the actual local C-space, where the first-order approximation works well when the rotational motion is small. Therefore, not only the inflation factor ϵ, but the roundness of the ellipsoid also affects the approximation. With the increase of the aspect ratio, which quantifies the roundness, the convex lower bound takes larger portion of the volume related to the actual local C-space.

8.3.1.2 Vertex Connections Based on the Convex Polyhedron Local C-space

To further connect two vertices, V_1 and V_2, by a collision-free path $Path_{12}$, one can first define a new middle vertex V_3 that is inside the intersection of the Local C-space of V_1 and V_2. Then connecting V_1, V_3 and V_2, V_3 by line segments gives $Path_{13}$ and $Path_{23}$ respectively. These two path segments are guaranteed to be collision-free since both are fully inside the convex polyhedron of V_1 or V_2. Finally, the desired collision-free path is a combination of the two segments, i.e. $Path_{12} = Path_{13} \cup Path_{23}$. Fig.8.5 demonstrates the proposed

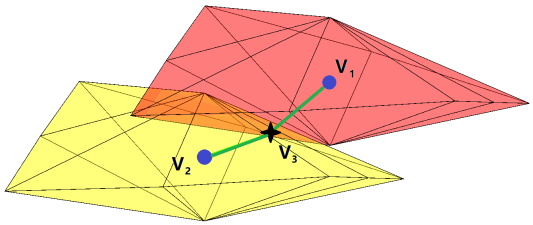


Figure 8.5: Edges between C-layers in the local C-space. Blue dots are the two vertices, V_1 and V_2, with convex polyhedron being their local C-space. The green line segments connect V_3 at the intersection and the two vertices respectively.

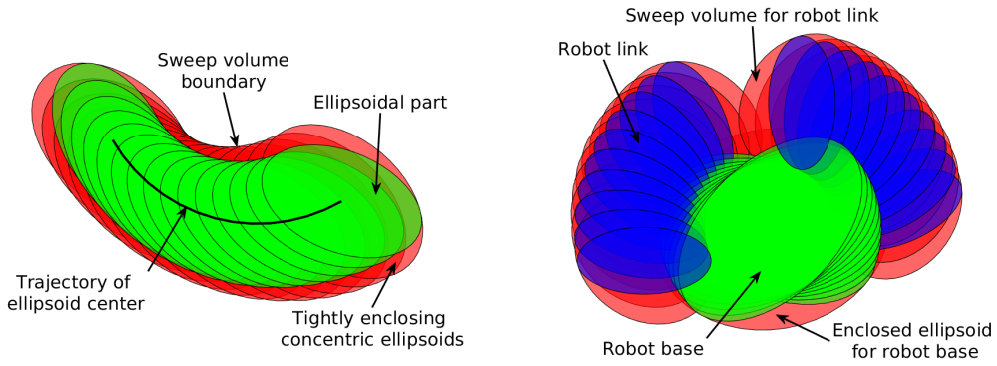
connection scheme for vertices in different C-layers.

8.3.2 Local planner 2: the Middle C-Layer

The local C-space idea works well when the two poses are close to each other (i.e. with the *small angle* assumption in the algebraic condition), especially in 2D. However, as shown in [124], when it comes to the 3D case, the local C-space occupies a much smaller portion of the total volume of the motion space, therefore making it difficult to enclose two poses if the orientation difference is large. In this subsection, a *middle C-layer* method is proposed to guarantee that the vertices at different C-layers can be safely connected, without the small angle assumption. The basic idea is to construct a sweep volume that tightly covers all the ellipsoidal parts at two poses based on the closed-form MVCE introduced in Sec. 3.2.2.

8.3.2.1 Sweep volume for individual ellipsoidal parts

Suppose that the part E_i is moving from vertex (pose) $V_1 = [\mathbf{t}_1^\top, \boldsymbol{\omega}_1^\top]^\top$ to $V_2 = [\mathbf{t}_2^\top, \boldsymbol{\omega}_2^\top]^\top$ while staying in the free space. The idea here is to bound E_i at the two poses by a tightly fitted sweep volume and ensure that it is fully



(a) Sweep volume for individual elliptical part.

(b) Sweep volume for the whole multi-link robot.

Figure 8.6: 2D example illustrating the sweep volume idea based on the sliding of tightly enclosed ellipsoids.

contained in the free space, then the motions within the sweep volume will be guaranteed safe. Given a trajectory connecting V_1 and V_2, the rotational parts of the intermediate steps can be retrieved. At first, E_i at these orientations are encapsulated by a concentric ellipsoid, E_{ci}, via the iterative computations of MVCE. Within this E_{ci}, transitions including rotations are allowed between the two end poses. Then, E_{ci} translates from $\mathbf{t}_1$ to $\mathbf{t}_2$ following the path of E_i's center. The resulting sweep volume bounds the transition of E_i between the two poses, subject to the pre-defined trajectory. Figure 8.6a shows the procedure of constructing the sweep volume.

To ensure that E_{ci} stays inside the collision-free space, Minkowski sums with obstacles are computed and the same sweep-line process is done to construct collision-free segmentations. Then if the path from $\mathbf{t}_1$ to $\mathbf{t}_2$ is within the collision-free segments, the sweep volume is guaranteed to be safe. Therefore, the whole transition for the ellipsoidal part E_i is collision-free.

8.3.2.2 Concatenation of the sweep volumes for poly-ellipsoidal robot

Once the sweep volumes of individual parts are calculated, the union of them covers the whole robot. If g_b represents the transformation of base ellipsoid E_1 as seen in the world frame, the transformation for each individual part is $g_b \circ g_i (i = 1, ..., M)$. Then, the translation component guides the sliding motion of the corresponding E_{ci}, which results in the sweep volume enclosing the motion of E_i. Figure 8.6b illustrates the sweep volume union that encloses the whole poly-ellipsoidal robot.

Similar to the process for individual part, the transition safety for the whole robot is guaranteed by computing Minkowski sums and querying the inclusion of the reference point in the free space. But now, the reference point of each individual part becomes its own center, not the center of the base as for the C-layer constructions. Therefore, each $E_{ci]}$ is treated independently and the collision-free C-space is constructed separately. If all the reference points are within their own free space, the union of sweep volume is collision free. This separative consideration of each sweep volume is necessary because relative rotations are involved between different parts, which means a unified reference point will not necessarily stay in the same C-layer.

8.3.2.3 Vertex connections based on middle C-layer calculations

In order to connect different C-layers efficiently, the motions of the robot are decomposed by pure rotations around the base ellipsoid center followed by translations. The proposed sweep volume is applied here to connect two vertices with different rotational parts, i.e. V_1 and V_2. At first, each

ellipsoidal part is enclosed by a tightly fitted concentric ellipsoid, based on the interpolated poses, and individual C-layer is constructed accordingly. Since the base ellipsoid is not translating, only the center reference point is queried inside the collision-free C-space. For the other parts, the paths of their centers around the base ellipsoid center are circular arcs, therefore, points on these arcs with respect to the world frame are computed and queried. Once all of the intermediate reference points are within their corresponding free C-space, the rotational motion is checked to be safe. Then an intermediate vertex $V_1' = [\mathbf{t}_1^\top, \boldsymbol{\omega}_2^\top]^\top$ is generated and added to the vertex list, and the edge between V_1 and V_1' is connected. Since a middle vertex is generated, the C-layers constructed for each tightly fitted concentric ellipsoid are called "middle C-layer", which can be thought of a bridge in the middle for the two vertices to be connected.

Now the robot is transformed into the C-layer where V_2 is in. Then, pure translation remains to be applied from V_1' to V_2, which locate in the same C-layer. In the current step, the safety of connections can be checked within the same C-layer. In addition, if the orientation samplings are incremental, there will not be a large rotational difference between adjacent C-layers. Therefore, connecting vertices on adjacent C-layers can be done efficiently, and the extra free space inside the sweep volume will be small.

8.4 The Hybrid Motion Planner using Closed-form Collision-free Configuration (C_F^3) Samples

Inspired by [92], a hybrid algorithm is proposed that generates a set of guaranteed collision-free configurations as follows. We first randomly sample configurations in $\mathrm{SO}(3) \times (S^1)^n$, each of which represents a random C-slice. At each C-slice, the orientation of the base part and joint angles are fixed, which defines a *shape* of the robot and the robot is treated as a rigid body with translation motions only. Then, a list of collision-free positions of the base in $\mathbb{R}^3$ is computed via the efficient closed-form Minkowski sums and a sweep-line process.

8.4.1 The hybrid algorithm to generate C_F^3 samples

Algorithm 3 illustrates the process of generating collision-free configuration samples.

The probabilistic steps are in Lines 1 and 7, which randomly generates the shape of the robot and z-coordinates on each collision-free line segments respectively. The first parameter of the "RandomGenerator" subroutine provides the configuration space to sample from, and the second defines the number of samples. Line 3 computes the closed-form Minkowski sums between the robot and obstacles, which results in an ordered list of discrete points on the C-obstacle boundaries, characterized according to angular parameters. Then, the surface meshes are easy to obtain from this ordered list of surface points. By pre-defined number of sweep lines along x and y axis, a list of (x, y) coordinates of the base part can be obtained in Line 4. Further, at

Algorithm 3: Procedure to generate C_F^3 samples on configuration space

Input: Robot geometry; obstacles and arena geometries and poses.
Parameters: Num. shapes (N_{shape}); Num. sweep lines (N_{line}); Num. points on sweep line (N_{point})
Output: List of C_F^3 samples

1 $\{(\mathbf{q}_{base}^T, \mathbf{q}_{link}^T)\} \longleftarrow$ RandomGenerator($\{\mathrm{SO}(3) \times (S^1)^n\}$, N_{shape});
2 **for** $(\mathbf{q}_{base,i}^T, \mathbf{q}_{link,i}^T)$ **do**
3 $C_{obstacle} \longleftarrow$ ClosedFormMinkowskiSum($Robot$, $Obstacles$) ;
4 $\{(x_{base}, y_{base})\} \longleftarrow$ DefineVerticalLines(N_{line});
5 **for** $(x_{base,j}, y_{base,j})$ **do**
6 $L_{CF} \longleftarrow$ SweepLineProcess($C_{obstacle}$, $\{(x_{base,j}, y_{base,j})\}$);
7 $\{z_{base}\} \longleftarrow$ RandomGenerator(L_{CF}, N_{point});
8 **for** $z_{base,k}$ **do**
9 $sampleList$.Append($\{x_{base,j}, y_{base,j}, z_{base,k}, \mathbf{q}_{base,i}^T, \mathbf{q}_{joint,i}^T\}$);
10 **end**
11 **end**
12 **end**

each line, the "SweepLineProcess" in Line 6 computes a set of collision-free line segments, i.e. L_{CF}. Finally, once the configuration is sampled randomly on each line segment, it is appended to the list of collision-free samples.

8.4.2 The motion planning framework using C_F^3 samples

The proposed C_F^3 generating algorithm provides a set of configuration sample seeds for probabilistic motion planners. The general framework for solving a motion planning problem is illustrated in this section and the implementation details are discussed.

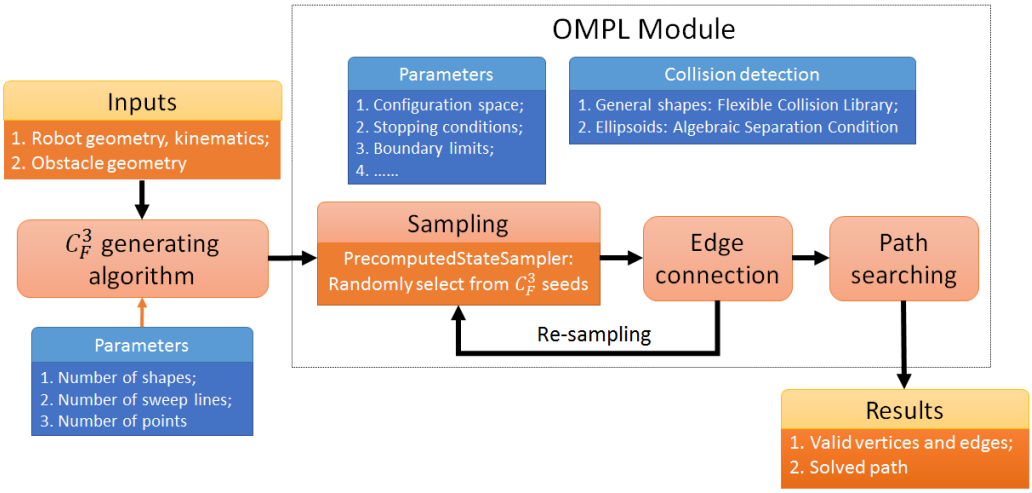


Figure 8.7: Workflow of the motion planning framework using C_F^3 samples as seeds for sampling-based algorithms.

8.4.2.1 Demonstration of the motion planning framework

A typical sampling-based motion planner includes free space exploration and valid edge connection steps, and our proposed C_F^3 samples can be treated as seeds before planning. At the planning phase, planners can randomly pick from this set of C_F^3 seeds and collision detection is applied just for validation and edge connection, which saves computational resources especially within a narrow area. Figure 8.7 demonstrates the workflow of out proposed C_F^3 generating step in solving a motion planning problem.

8.4.2.2 Implementation details

The implementation is based on the framework of OMPL [142], And the generated set of samples can be fed into any planner that are available in this popular library. In the sample generating phase, the discrete points on Minkowski sums boundary of each obstacle are first computed. 100 discrete

points are chosen as representations of each C-obstacle. Afterwards, for calculations of intersecting points in practice, a surface mesh is generated based on this set of points, which can be performed efficiently since the point set is organized based on the angular parameters. Then, during the sweep-line process, a line-mesh intersection procedure is used to compute intersecting points between each sweep line and C-obstacles. Once the C_F^3 samples are generated, the class *PrecomputedStateSampler* from OMPL's *base* namespace is used to pick a configuration at random from the generated list of samples during the planning phase.

Several pre-defined parameters affect the performance of our proposed algorithm: (1) The number of robot shapes (N_{shape}) specifies the internal configurations of the robot and the orientations of the robot base; (2) The number of sweep lines (N_{line}) and points on each collision-free line segment (N_{point}) determine the amount of space to be explored before the planning phase.

8.5 Benchmarks 1: Geometric Rigid-body Planning for Poly-Ellipsoidal Robots

The proposed Highway RoadMap planner is able to deal with both $\mathrm{SE}(2)$ and $\mathrm{SE}(3)$ rigid body geometric path planning problems, where the robot is constructed by multiple rigidly connected 2D ellipses and 3D ellipsoids respectively. As a demonstration of its capabilities, $\mathrm{SE}(2)$ path planning examples are shown at first. Then, to further evaluate its performance, benchmarks

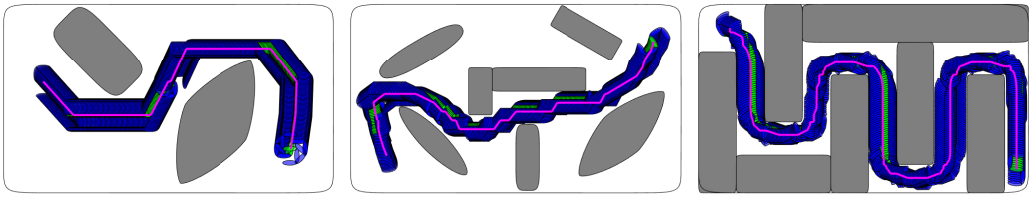
Table 8.1: Parameters explanations for the experiments.

Parameters	Explanations
N_{Layers}	Number of C-layers for Highway RoadMap
N_{Sweep}	Number of sweep lines for Highway RoadMap
N_{Vertex}	Number of valid configurations on the graph
N_{Edge}	Number of valid edges
N_{Path}	Number of vertices on the solved path

are conducted with some famous sampling-based planners in the SE(3) rigid-body planning problems, where the robot is now enclosed by a union of ellipsoids and the obstacles are modeled as superquadrics. The planner is written in C++ and all the benchmarks are ran on an Intel Core i7 CPU at 3.40 GHz. For the following experiments, several parameters are examined and a list of them with explanations is summarized in Table 8.1.

8.5.1 Demonstrations of SE(2) rigid body path planning for poly-elliptical robots

For this set of experiments on SE(2), the robot is modeled as an S-shape, which is constructed as a movable base at the center and 4 links being serially and rigidly attached. Three kinds of environments are considered and path planning problems are solved by our proposed planner. Fig. 8.8 shows these SE(2) planning scenarios in sparse, cluttered and dense environments respectively. The number of C-layers and horizontal sweep lines are pre-defined according to the sparsity of the environment. Tables 8.2 displays the parameters for the roadmap construction and the solution path after the graph search process. The planning time is also measured, which includes both roadmap construction and graph search time, as in Tab. 8.3. The results



(a) Sparse map **(b)** Cluttered map **(c)** Maze map

Figure 8.8: Demonstration of the capability of our algorithm for planning valid paths in different kinds of environments. Obstacles are superellipses and the robots are constructed as an S-shape. The magenta curve represents the trajectory of the robot base ellipse.

Table 8.2: Roadmap and solution path information for $\mathrm{SE}(2)$ planning experiments using HighwayRoadMap framework.

Map	N_{Layers}	N_{Sweep}	N_{Vertex}	N_{Edge}	N_{Path}
Sparse	5	6	110	163	13
Cluttered	20	20	1767	2828	45
Maze	30	40	2890	4698	189

Table 8.3: Planning time for $\mathrm{SE}(2)$ experiments using HighwayRoadMap framework.

Map	Search time	Total time
Sparse	0.071564 ms	6.63451 ms
Cluttered	0.909632 ms	61.0353 ms
Maze	1.54588 ms	106.267 ms

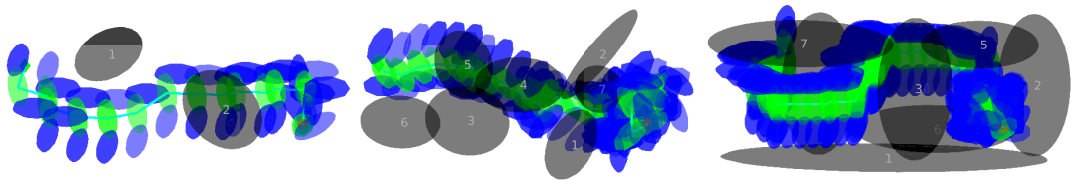
for all the scenarios are around the level of milliseconds, which indicates the effectiveness of the proposed framework in solving 2D rigid body planning problems, even in the maze environment with many narrow corridors.

8.5.2 Benchmarks for SE(3) planning problems

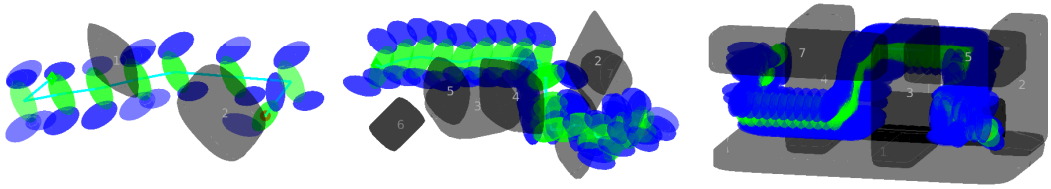
The performance of our proposed framework is further evaluated by benchmarking with several widely-used sampling-based planners from the well-known OMPL, i.e. PRM [77], Lazy PRM [18], RRT [86], RRT Connect [82]

and EST [67]. Moreover, different sampling methods that enhance the effectiveness for probabilistic planners are also compared, including uniform random sampling (Uniform), obstacle-based sampling (OB) [123], Gaussian sampling (Gaussian) [19], bridge test (Bridge) [66] and maximized clearance sampling (Max Clearance). Similarly to the $\mathrm{SE}(2)$ cases, three map types are considered: a sparse map, where there are only two obstacles and the free space occupies a majority of the area; a cluttered map, where more space is occupied by obstacles which are placed in different random orientations and positions; and a maze map, where only some narrow corridors are available to pass. The robot is constructed by a free-floating base with two links being rigidly attached on the side.

50 planning trials are ran per planner per map, and a time limit of 60 seconds is set for one planning trial, which means that the planning fails if the time exceeds this limit. As for the solution, the running time is compared for solving each problem and the success rate among multiple experimental trials. Also, the number of vertices and edges being generated, and the number of vertices on the solved path are examined for all algorithms. As an important special example in the superquadrics family, an ellipsoidal model is first applied as a representation of obstacles. Then the general convex superquadric model is used for the experiments to represent different kinds of geometric shapes. Figure 8.9 shows these maps used for benchmark as well as a valid path being solved by Highway RoadMap algorithm.



(a) Ellipsoids, sparse **(b)** Ellipsoids, cluttered **(c)** Ellipsoids, maze



(d) Superquadrics, sparse **(e)** Superquadrics, cluttered **(f)** Superquadrics, maze

Figure 8.9: The maps with superquadric obstacles for benchmarking the algorithms. The robot is a union of three ellipsoids being rigidly connected.

8.5.2.1 Benchmark settings

For the proposed algorithm, each C-layer is defined by a sampled orientation. And specifically, the rotational symmetries of the icosahedron [158] are chosen, which consists of 60 elements of $SO(3)$ and forms a finite subgroup. Note that more rotations can be sampled to construct a denser roadmap. Substantial literature exists for sampling rotations on $SO(3)$ [6], [163], [167], either randomly or deterministically, each of which has advantages on different aspects.

For sampling-based planning algorithms, the majority of time is consumed in collision checking, so the choice of a relatively fast collision checker is a priority. The open-source and widely-used library, Flexible Collision Library (FCL) [116], is chosen as an external plug-in for sampling-based planners. In particular, a special and efficient collision object from FCL is applied for

ellipsoids, where 12 vertices are pre-defined to bound the exact ellipsoidal surface. The exact solver for determining ellipsoid separations, i.e. Algebraic Separation Conditions for ellipsoids (ASC) [152], is implemented and applied in the benchmark. Furthermore, for superquadrics, their surfaces are discretized as triangular meshes based on the parametric expressions (the body that this discrete surface encloses can be seen as a convex polyhedron), and a bounding volume hierarchy for oriented bounding boxes are used as the geometric primitive, which is a common efficient collision detection technique implemented in FCL. Since the efficiency and accuracy of collision checking highly depend on the quality of discretization, to make the comparison relatively fair, a statistical evaluation is provided to determine the numbers of vertices for the discrete superquadric surface.

To quantify the quality of discretization, relative values between the volume of the discrete polyhedron approximation and the exact volume are computed, i.e.

$$\kappa = \frac{Vol_{poly}}{Vol_{SQ}}, \tag{8.2}$$

where,

$$Vol_{SQ} = 2a_1a_2a_3\epsilon_1\epsilon_2\beta(\frac{\epsilon_1}{2}+1,\epsilon_1)\beta(\frac{\epsilon_2}{2},\frac{\epsilon_2}{2}), \tag{8.3}$$

where a_1, a_2, a_3 are the semi-axes lengths, ϵ_1, ϵ_2 are the exponents and $\beta(\cdot)$ is the Beta function, i.e. $\beta(x,y) = 2\int_0^{\pi/2} \sin^{2x-1}\phi \cos^{2y-1}\phi \; d\phi$. The relative volumes are computed for different numbers of vertices on the surface, i.e. from 16 to 400. Then for each discretization number, 50 different superquadric shapes are randomly generated by changing the parameters such as semi-axes

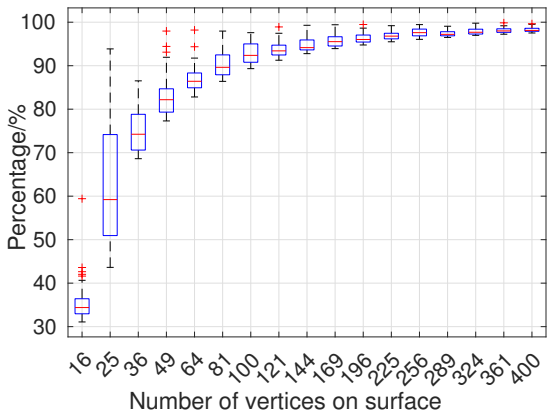


Figure 8.10: Relative volume evaluations for discrete approximated superquadrics.

lengths and exponents. Figure 8.10 shows the statistical plot of the discretization quality for different vertex resolutions. After around 100 vertices, the relative volume starts to be plateaued and above 90%. Therefore, for the experiments, 100 is chosen as the number of vertices for the superquadric surface.

8.5.2.2 Parameters for Highway RoadMap planner

Table 8.4 provides the implementation details for the Highway RoadMap planner, where the number of C-layers and sweep lines on each layer are predefined according to the sparsity of the environment. Note that the number of sweep lines on each C-layer is a multiplication of the numbers of lines along x and y axes. Therefore, $N_{SL}(N_x \times N_y)$ is displayed in the table.

8.5.2.3 Benchmark results

In this part, the benchmark results are shown between the proposed Highway RoadMap planner and some popular sampling-based planners. Comparisons

Table 8.4: Parameters for Highway RoadMap planner

Shape	Map	N_{Layers}	N_{Sweep}
Ellipsoids	Sparse	60	15 (5×3)
Ellipsoids	Cluttered	60	150 (15×10)
Ellipsoids	Maze	60	228 (19×12)
Superquadrics	Sparse	60	15 (5×3)
Superquadrics	Cluttered	60	150 (15×10)
Superquadrics	Maze	60	352 (22×16)

include: the resulting numbers of vertices (N_{Vertex}) and edges (N_{Edge}) on the graph/tree structures, the number of vertices on the solution paths (N_{Path}), the planning time and success rate to find a valid path.

The resulting numbers of vertices and edges information are averaged and rounded to the nearest integer. Tables 8.5 and 8.6 show the benchmark parameters comparisons with the sampling-based planners for environment with ellipsoidal and superquadric obstacles respectively. As can be viewed from these numbers, our Highway RoadMap planner generates relatively more collision-free vertices and edges than sampling-based planners, therefore the resulting solution path has more via points. Compared to the graph-based probabilistic planners, which connect a vertex with its several nearest neighbors within a range, Highway RoadMap only connects vertices in adjacent sweep lines at each C-layer. And its graph size is determined by the predefined parameters, i.e. the number of C-layers, sweep line resolution and the complexity of the environment.

The comparisons for total planning time and success rate is shown in Fig. 8.11 in both ellipsoid and superquadric obstacles cases. Since the proposed

Table 8.5: Benchmark parameters for ellipsoidal obstacles case between Highway RoadMap and sampling-based planners

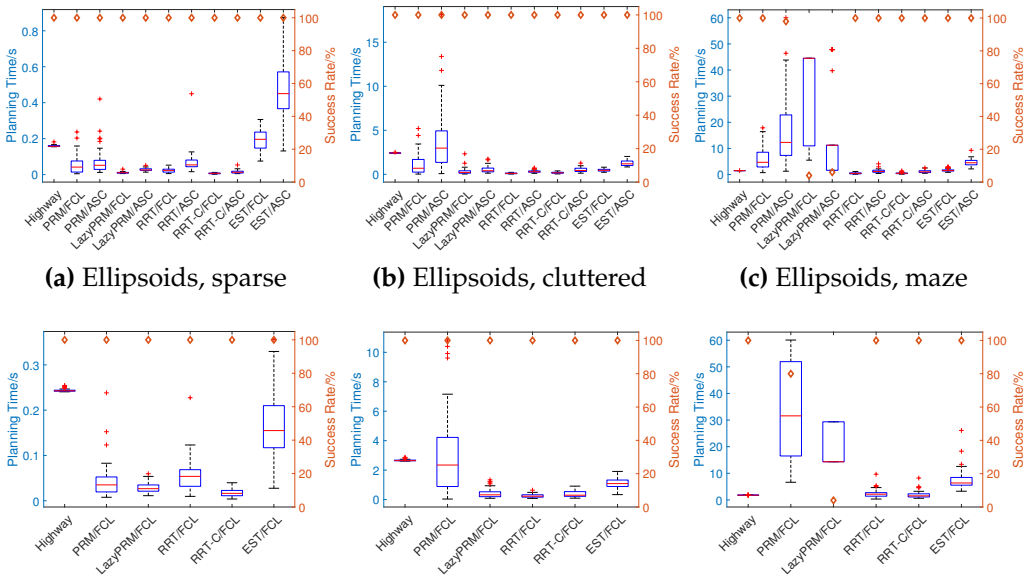
Map	Planner	Collision	N_{Vertex}	N_{Edge}	N_{Path}
Sparse	Highway	–	1250	1802	25
Sparse	PRM	FCL	18	140	5
Sparse	PRM	ASC	12	70	5
Sparse	Lazy PRM	FCL	53	313	11
Sparse	Lazy PRM	ASC	53	306	11
Sparse	RRT	FCL	34	33	9
Sparse	RRT	ASC	38	37	9
Sparse	RRT Connect	FCL	13	12	9
Sparse	RRT Connect	ASC	12	11	8
Sparse	EST	FCL	230	229	15
Sparse	EST	ASC	213	212	14
Cluttered	Highway	–	11812	17686	43
Cluttered	PRM	FCL	196	1268	9
Cluttered	PRM	ASC	260	1634	9
Cluttered	Lazy PRM	FCL	871	6900	18
Cluttered	Lazy PRM	ASC	805	6350	18
Cluttered	RRT	FCL	47	46	10
Cluttered	RRT	ASC	51	50	11
Cluttered	RRT Connect	FCL	78	77	12
Cluttered	RRT Connect	ASC	79	78	12
Cluttered	EST	FCL	170	169	15
Cluttered	EST	ASC	164	163	15
Maze	Highway	–	5938	7635	85
Maze	PRM	FCL	1095	8078	24
Maze	PRM	ASC	1076	7874	25
Maze	Lazy PRM	FCL	5741	4699	46
Maze	Lazy PRM	ASC	3195	2575	36
Maze	RRT	FCL	190	189	17
Maze	RRT	ASC	197	196	17
Maze	RRT Connect	FCL	197	196	18
Maze	RRT Connect	ASC	175	174	18
Maze	EST	FCL	345	344	26
Maze	EST	ASC	367	366	28

planner constructs a graph structure and can answer multiple queries, PRM-based planners with different types of samplers are specifically picked as

Table 8.6: Benchmark parameters superquadric obstacles case between Highway RoadMap and sampling-based planners

Map	Planner	Collision	N_{Vertex}	N_{Edge}	N_{Path}
Sparse	Highway	–	1297	1818	9
Sparse	PRM	FCL	10	40	5
Sparse	Lazy PRM	FCL	54	321	12
Sparse	RRT	FCL	31	30	9
Sparse	RRT Connect	FCL	12	11	8
Sparse	EST	FCL	81	80	9
Cluttered	Highway	–	11627	18279	48
Cluttered	PRM	FCL	154	1026	9
Cluttered	Lazy PRM	FCL	594	4679	16
Cluttered	RRT	FCL	38	37	10
Cluttered	RRT Connect	FCL	63	62	10
Cluttered	EST	FCL	134	133	14
Maze	Highway	–	5276	7500	62
Maze	PRM	FCL	1374	10734	19
Maze	Lazy PRM	FCL	6716	50336	64
Maze	RRT	FCL	318	317	19
Maze	RRT Connect	FCL	276	275	20
Maze	EST	FCL	425	424	28

the comparison objectives in Fig. 8.12. Although in the sparse map, our Highway RoadMap is relatively slower than the probabilistic planners, as the map becomes more dense and cluttered, its speed starts to take the lead among other planners. It becomes more obvious in the maze map that the speed of Highway RoadMap is much faster than the PRM-based planners, and competitive with RRT-based ones. Furthermore, because of the deterministic nature, Highway RoadMap behaves much more stable among different trials, while sampling-based planners have relatively large variance.



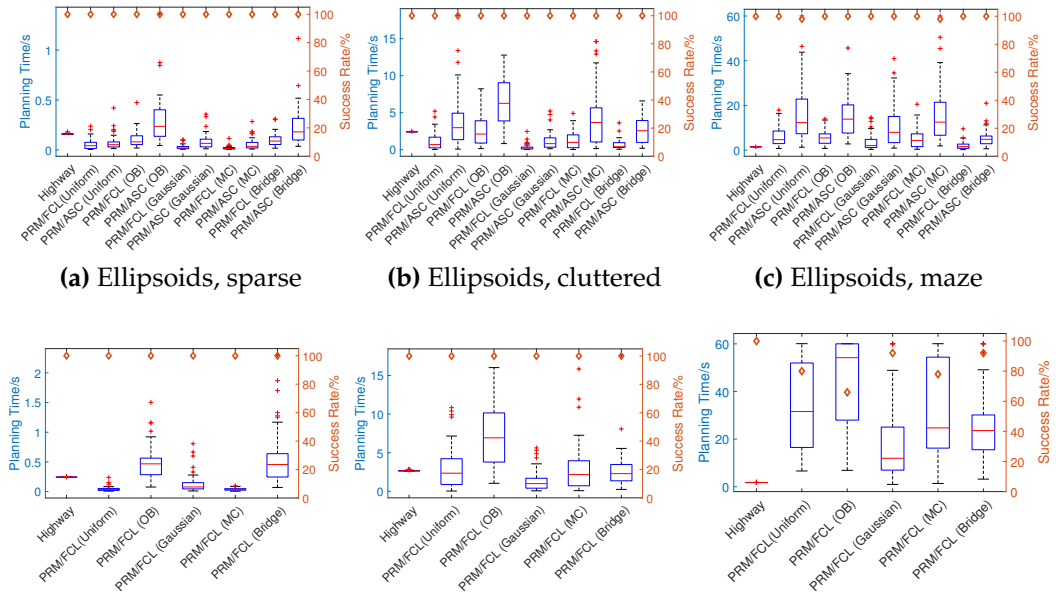
(a) Ellipsoids, sparse **(b)** Ellipsoids, cluttered **(c)** Ellipsoids, maze

(d) Superquadrics, sparse **(e)** Superquadrics, cluttered **(f)** Superquadrics, maze

Figure 8.11: Planning time and success rate comparisons with sampled-based motion planners. Planners' labels are formatted as "planner name/collision checker".

8.6 Benchmarks 2: High Dimensional Planning for Articulated Robots

Benchmarks for the proposed hybrid C_F^3 generating algorithm are conducted in high dimensional path planning problems. Comparisons are made between the sampling-based planners from OMPL and the same ones treating our proposed C_F^3 samples generated as prior knowledge. The benchmarks include planning in 2D and 3D workspace in cluttered environments with narrow passages. The robot is considered as either rigid-body or articulated-body. For each case, 50 trials are ran and statistical results are collected. The implementations are written in C++ under OMPL framework, and all the benchmarks are ran on an Intel Core i7 CPU at 3.40 GHz. Figure 8.13 shows the planning



(a) Ellipsoids, sparse **(b)** Ellipsoids, cluttered **(c)** Ellipsoids, maze

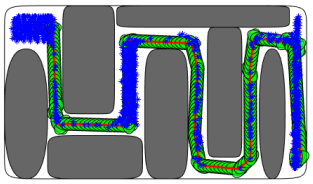
(d) Superquadrics, sparse **(e)** Superquadrics, cluttered **(f)** Superquadrics, maze

Figure 8.12: Planning time and success rate comparisons with PRM using different sampling methods. Planners' labels are formatted as "planner name/collision checker (sampler)".

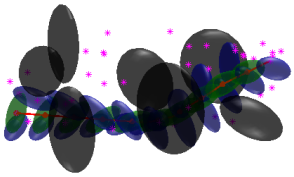
scenes and results of the proposed method. The dimensions of problems varies from 3 to 15; and the motion primitives include free-flying and first-order car models.

8.6.1 Parameters for C_F^3 sample generations

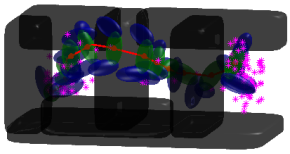
The parameters to compute the list of C_F^3 samples are pre-defined for each experiment case, and are summarized in Tab. 8.7. For each case, several parameters are shown: the dimension and type of the work space, the shape of the obstacles, the dimension of the configuration space, the type of motion primitives that the robot is required to follow, the number of shapes to be sampled (N_{shape}), the number of sweep lines to be generated (N_{line}) and the



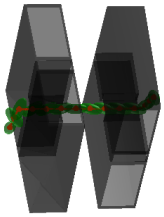
(a) SE(2), Dubins, 3-DOF; maze, superellipses



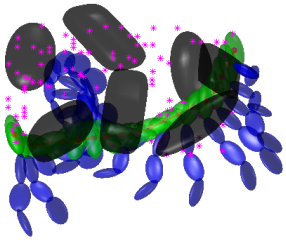
(b) SE(3), 6-DOF; cluttered, ellipsoids



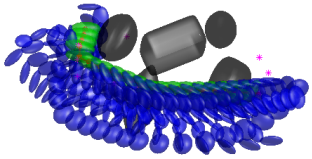
(c) SE(3), 6-DOF; maze, superquadrics



(d) SE(3), 6-DOF; narrow, superquadrics



(e) SE(3) $\times$ $(S^1)^3$, 9-DOF; cluttered, superquadrics



(f) SE(3) $\times$ $(S^1)^9$, 15-DOF; cluttered, superquadrics

Figure 8.13: Planning scenes for benchmark and the paths generated by using C_F^3 samples and RRT-connect algorithm. The obstacles, robot base, and robot links are shown in black, green, and blue, respectively. Configuration space information is noted on each sub-caption. The planned paths are shown as bold line segments. The sampled configurations projected to the workspace in the planning phase are plotted as asterisks.

Table 8.7: Parameters list to compute C_F^3 samples.

Map	Obstacles	DOF	Motion	N_{shape}	N_{line}	N_{point}
2D maze	superellipses	3	Dubins	50	50	20
3D cluttered	ellipsoids	6	Free-flying	60	900	20
3D maze	superquadrics	6	Free-flying	60	900	20
3D narrow	superquadrics	6	Free-flying	60	900	20
3D cluttered	superquadrics	9	Free-flying	60	900	20
3D cluttered	superquadrics	15	Free-flying	60	400	10

number of points to be sampled on each collision-free line segment (N_{point}). Tab. 8.8 shows the resulting number of C_F^3 samples (N_{sample}) and the averaged running time to compute these samples.

Table 8.8: Number of C_F^3 samples and computation time.

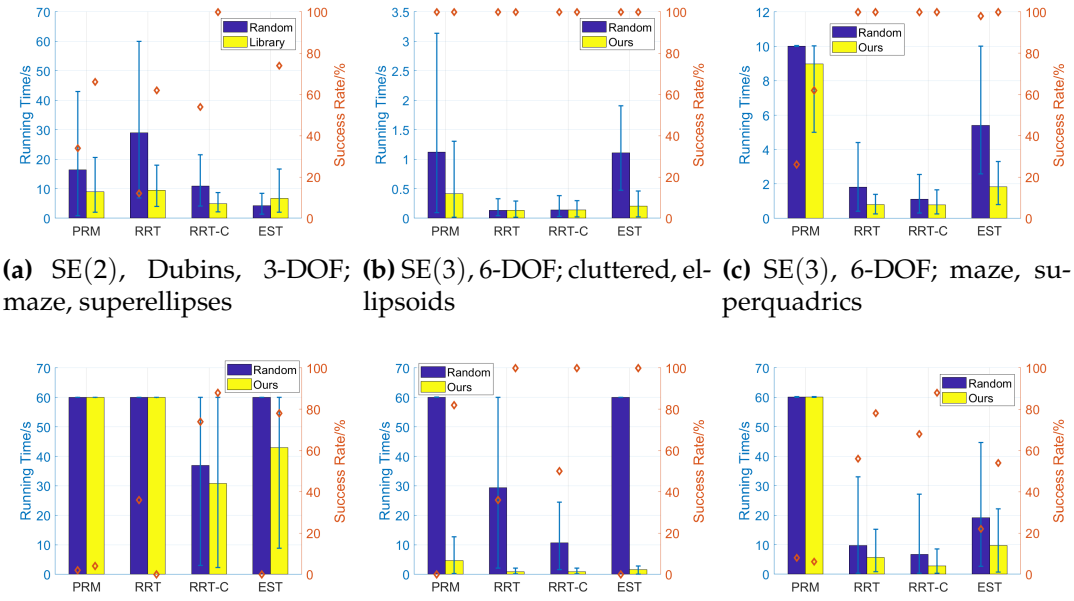
Map	Obstacles	DOF	Motion	N_{sample}	Time
2D maze	superellipses	3	Dubins	164700	0.4174s
3D cluttered	ellipsoids	6	Free-flying	1086260	2.1493s
3D maze	superquadrics	6	Free-flying	241380	2.5018s
3D narrow	superquadrics	6	Free-flying	564680	0.9597s
3D cluttered	superquadrics	9	Free-flying	917340	6.6827s
3D cluttered	superquadrics	15	Free-flying	148000	6.7670s

8.6.2 Benchmark results

The running time of the planning phase and the success rates are the two comparison metrics. Since our proposed algorithm is able to provide free sample seeds for any planner in OMPL, benchmarks are conducted with four representative planners, including both multi-query and single-query ones, i.e. PRM [77], RRT [86], RRT-Connect [82] and EST [67]. The sampling method for the vanilla probabilistic planners is set as uniform random. In this set of benchmarks, uniformly random sampling is used as the sampling method for the vanilla probabilistic counterparts. The results are shown in Fig.8.14. Furthermore, specifically for PRM, different online biased sampling methods are also compared, i.e. Gaussian [19], obstacle-base [3], max-clearance and bridge-test [66] (Fig. 8.15).

8.7 Physical Experiments on Walking Path Planning for Elliptical Projection of a Humanoid Robot

In this section, physical experiments are conducted on a path planning problem for a NAO humanoid robot. The task is to guide the robot to walk through



(a) $\mathrm{SE}(2)$, Dubins, 3-DOF; maze, superellipses **(b)** $\mathrm{SE}(3)$, 6-DOF; cluttered, ellipsoids **(c)** $\mathrm{SE}(3)$, 6-DOF; maze, superquadrics

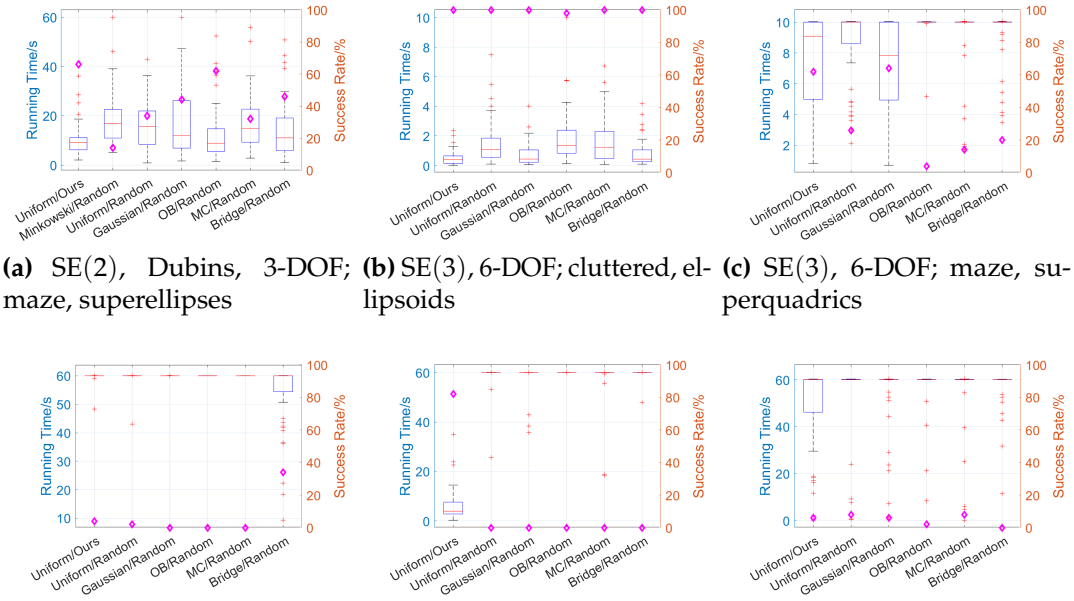
(d) $\mathrm{SE}(3)$, 6-DOF; narrow, superquadrics **(e)** $\mathrm{SE}(3) \times (S^1)^3$, 9-DOF; cluttered, superquadrics **(f)** $\mathrm{SE}(3) \times (S^1)^3$, 15-DOF; cluttered, superquadrics

Figure 8.14: Running time and success rate comparisons between planners using uniformly random sampling and our C_F^3 generator in solving different planning problems. The height of each bar indicates the averaged running time, and the end points of the blue line segment shows the maximum and minimum running time among the 50 trials. The diamond points are the corresponding success rates of different planners.

an environment with several boxes in random poses. The robot is not required to step on top of the boxes, but just avoid them in order to pass this cluttered space. Therefore, the problem is simplified into a planar case, where the contour of the robot from the top view is encapsulated by an ellipse.

8.7.1 Experiment setups

Assume that the dimensions of arena and boxes are known a priori, and the modeling process is described as follows. The arena that encloses all the obstacles and the allowable space for the robot to move is defined as



(a) $\mathrm{SE}(2)$, Dubins, 3-DOF; maze, superellipses

(b) $\mathrm{SE}(3)$, 6-DOF; cluttered, ellipsoids

(c) $\mathrm{SE}(3)$, 6-DOF; maze, superquadrics

(d) $\mathrm{SE}(3)$, 6-DOF; narrow, superquadrics

(e) $\mathrm{SE}(3) \times (S^1)^3$, 9-DOF; cluttered, superquadrics

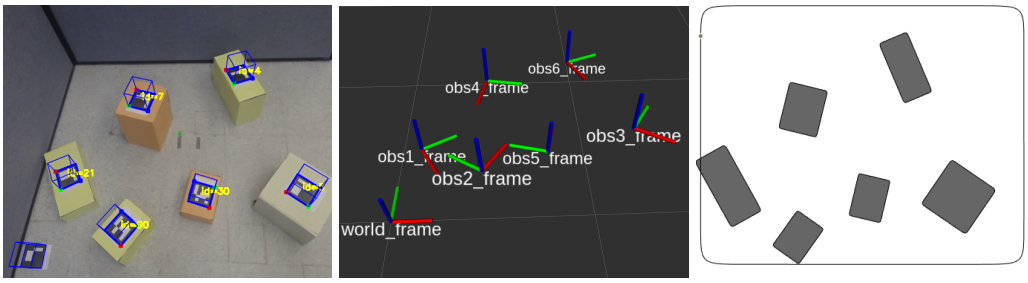
(f) $\mathrm{SE}(3) \times (S^1)^3$, 15-DOF; cluttered, superquadrics

Figure 8.15: Running time and success rate comparisons between different valid state samplers in solving the planning problems in different environments. The box plot shows the statistics of the running time benchmark experiments, and the diamond points are the corresponding success rates of the experiment cases.

a rectangle area, which is bounded by a superellipse with $\epsilon = 0.1$. And the obstacles are rectangular boxes placed in random poses, therefore they are enclosed as superellipses with $\epsilon = 0.1$. The sizes of these boxes are manually measured to determine the semi-axes lengths of the corresponding enclosed superellipses. The world reference frame and the local frame of each obstacle are indicated by *ArUco* tags attached on their top surfaces, which can be recognized by computer vision algorithms [53]. For the robot, its projection contour is encapsulated from the top view as an ellipse, which is conservative to bound all possible walking poses. This setup is reasonable since in household or office environments, furnitures have fixed size and

Table 8.9: Pre-defined shapes of the environment and elliptical encapsulation of the robot projection.

Object	Semi-axes lengths (centimeters)	Exponent
Arena	75, 60	0.1
Obstacle 1	10, 7.5	0.1
Obstacle 2	10, 7.5	0.1
Obstacle 3	12.5, 12.5	0.1
Obstacle 4	15, 7	0.1
Obstacle 5	17.5, 8	0.1
Obstacle 6	11, 9	0.1
Robot	12.5, 17.5	1



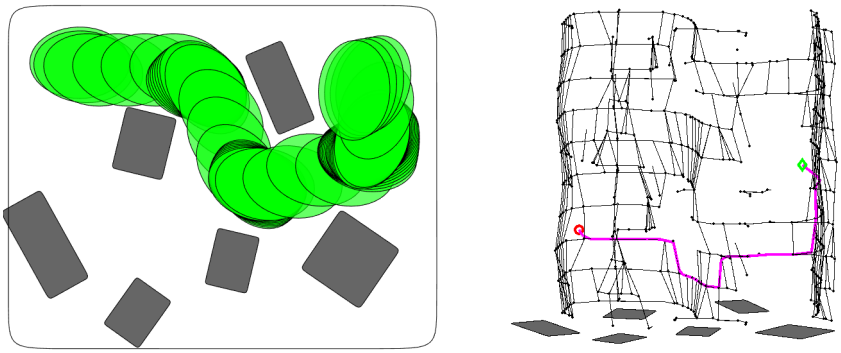
(a) Recognized ArUco tags. **(b)** Frames of obstacles. **(c)** Reconstructed environment.

Figure 8.16: The environment setups: ArUco tags are attached at the center of obstacles, which is recognized by a camera. The obstacle frames relative to the world frame is then computed and input into the planner.

only their poses are not known in advance. Table 8.9 shows the shapes of the predefined environment and robot, which are obtained off-line before planning. Figure 8.16 shows the planning environment constructed by sensing the frames of the obstacles.

8.7.2 Results

By setting the start and goal poses for the robot, the SE(2) planner of the Highway RoadMap framework is used to plan a path for the robot to go



(a) Solved path in simulation

(b) Visualization of the constructed graph structure and solved path

Figure 8.17: Path planned using Highway RoadMap framework. The solution and the graph structure with vertices and edges are visualized in simulation. Start and end poses are plotted as a red circle and a green diamond respectively; vertices are shown in black dots and connected by line segments; the solved path on the graph is shown in bold magenta line segments.

Table 8.10: Parameters and planning solution details for the real experiment.

Start/Goal	N_{layers}	N_{sweep}	N_{Vertex}	N_{Edge}	N_{Path}	Time
$[25, 100, -\frac{\pi}{2}]$, $[120, 90, 0]$	10	20	524	810	23	6.2713 ms

through this cluttered environment. The solved path requires the robot to walk through a narrow space, which is shown in simulation with the graph structure visualized in Fig. 8.17. Figure shows a sequence of snapshots from the real experiment. The parameters for solving this problem and the solution details are summarized in Tab. 8.10. Figure 8.18 demonstrates the real planning scenario with the NAO humanoid robot following the SE(2) path solved by our Highway RoadMap planner.

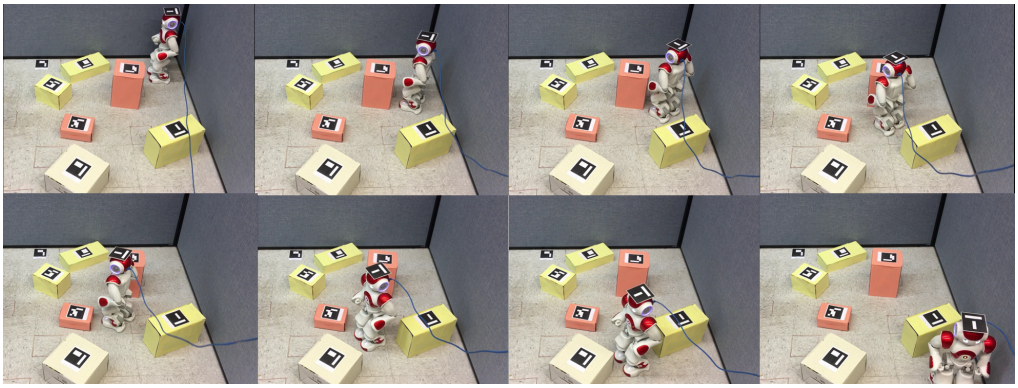


Figure 8.18: Snapshots for the path planning demonstration on a NAO humanoid robot.

8.8 Discussion

From the benchmark results, the proposed Highway RoadMap planner is able to efficiently solve geometric rigid-body path planning problems with multi-body robot in different scenarios. One of the highlights of this algorithm is the closed-form parameterization of Minkowski sum and difference that explicitly characterizes the C-space. In addition, the effectiveness of the sweep line method on a single C-layer provides a way to avoid traditional collision detection computation in generating collision-free samples, especially in narrow passage regions.

The two proposed local planners give connection strategies for vertices on adjacent C-layers. The idea of *local C-space* requires the existence of the intersection volume between the local C-space of the two vertices. A necessary condition for this existence requirement is that the *gaps* between adjacent C-layers should be small, therefore there is a trade-off between selecting the inflation factor and the number of layers. On the other hand, the idea of *middle*

C-layer firstly computes enclosed voids for different parts of the robot in closed form, using the similar fashions as in the Minkowski sum calculations. The idea of sliding the void to generate sweep volume restricts its motions into translation only, which reduces the dimension of the problem.

Combining the graph information and running time results, Highway RoadMap has the ability to generate more collision-free configurations in a relatively short time. At first, vertices generated in each C-layer are automatically guaranteed to be safe because they are all created outside the boundaries of C-obstacles. Then, the sweep volume of a tightly enclosed ellipsoid transfers an $\mathrm{SE}(3)$ sequence of each body into an $\mathbb{R}^3$ path of the enclosed void. This procedure constructs another C-layer for the enclosed voids between the two adjacent C-layers of the robot, which doubles the number of C-layers, but reduces the dimension when connecting two vertices. The enclosed ellipsoid and the corresponding sweep volume generate a conservative upper bound of the possible motions of each body, but their tightnesses make the extra space as small as possible. Moreover, the deterministic nature makes the speed and success rate of the proposed algorithm stable over different trials of the experiments on problems with up to six degrees of freedom.

On the other hand, sampling-based planners are very efficient when the environment is sparse, but start to drop speed as the space occupied by obstacles increases. This is because most of the sampled configurations are discarded. Therefore, the *sample-check-discard-resample* process iterates much longer than in the sparse maps. Also, the success rates are decreasing when the environment becomes denser. Especially for graph-based algorithms, even

with the help of effective biased samplers, a large amount of random samples are still discarded in the maze map. The tree-based planners are more efficient in sparse and cluttered maps, which have high chances of success and fast speed in single queries. But similarly, in the maze map, both the speed and success rate start to drop significantly.

Since a majority amount of computational resources are used in explicit collision detection between the robot parts and obstacles, the effectiveness of sampling-based planners depends highly on the collision checkers and the quality of surface discretization. To make the comparisons relatively fair, a widely recognized open-source collision library, FCL, is picked as a plug-in to the sampling-based planners, as well as an efficient exact collision detection method, ASC, specially designed for ellipsoids. For the special case of ellipsoid fitted environment, collision checking via both FCL and ASC are applied and the planning results are compared. It has been shown that the probabilistic planners run much faster when obstacles are modeled as ellipsoids. Even with the speed-up in this important special case, in the dense maze map, our proposed planner can still be around the order of twice faster compared with graph-based planners. And for the general superquadric obstacles, the number of vertice on the discrete surface is firstly determined by comparing relative volume with the exact value and generate meshes beforehand. The benchmark results show that, in the maze map, ours run around an order of 10 times faster than the graph-based planners, and around the same level with the tree-based ones. Consider the property that ours is a multi-query planner, such a computational speed is advantageous in solving rigid body planning

problems, especially in the narrow passage challenge.

One major limitation of the proposed Highway RoadMap planner, however, is its extension to solve high degree-of-freedom problems for articulated robots, which is not yet clear. The computational complexity of a naive implementation would grow exponentially as the dimension of C-space increases. The current algorithm is effective in dealing with rigid-body robot that has translation motions, but not suitable for a robot with fixed base such as manipulators. In the case of pure rotational motions, sampling is still required, and the advantage of closed-form Minkowski sum will not be obvious. Therefore, a hybrid planning algorithm is proposed by efficiently computing collision-free configurations a priori, where the rotational parts are sampled and configuration space obstacles can then be generated for each combination of the joints rotations.

With the proposed C_F^3 samples being treated as seeds, probabilistic motion planners can achieve more efficient performance in solving difficult problems such as the *narrow passage* challenge. The proposed algorithm parameterizes the collision-free configuration space via the closed-form Minkowski sums, and uses an efficient sweep line process to detect the free portion of the whole slice of the C-space. Its hybrid nature, in the sense that the robot shapes are sampled randomly and the free configurations in each C-slice are generated deterministically, provides an effective way to deal with high dimensional problems.

For each new setting of the problem, the C_F^3 samples are generated, allowing prior knowledge of the free space to be discovered before running the

planning solvers. The generated set of samples can be used multiple times during the planning phase, which enables the planner to choose the guaranteed collision-free samples without further *sample-check-discard-resample* iterations to find a feasible configuration. From the experimental results (Tab. 8.7), the number of C_F^3 samples can be generated at a speed ranging from 10,000 to 100,000 samples per seconds, depending on the pre-defined parameters, the dimension of the configuration space and the complexity of the environment. With the guidance of this generated set of collision-free samples, planners can quickly explore the map and return a valid feasible path much faster compared to using the default uniformly random sampling method.

Benchmark results show that, in almost all cases of 2D and 3D workspace, planners using our C_F^3 samples outperform the ones using the default uniformly random sampling method in terms of both the running speed and success rate. The advantages of our method become more significant when the environment becomes denser with more obstacles and narrow passages, e.g., 2D maze maps and 3D narrow maps. Also, when the dimension of the configuration space increases, planners using the pre-computed set of C_F^3 sample seeds achieve much higher chances of success and faster planning speed than using the uniformly random sampling method. Moreover, when comparing with PRM using biased online sampling methods, selecting configurations from our C_F^3 set also provides more promising results, especially for the cases of dense environments with narrow passages. However, in some cases, samplers including Gaussian and bridge-test are more effective because they keep samples that are close to obstacles, allowing more feasible configurations to be

explored inside the narrow area. But when the dimension becomes higher, our method starts to performs better while these online samplers stuck in some narrow areas that are difficult to connect a valid path between the sampled configurations.

The limitations of our proposed algorithm are stated as follows. Firstly, the C_F^3 samples are generated before the main planning algorithm, and the parameters are all pre-defined, which makes the free space explorations less robust. Also, due to the hybrid nature, the robot shapes is firstly sampled randomly. Therefore, when the robot base is fixed or cannot translate (e.g., manipulators), our algorithm is degraded into a vanilla sampling-based method. To further accommodate these limitations, future works include updating the planning parameters (e.g. the number of sampled shapes of the robot and the number of sweep lines) adaptively during the planning process to tackle the robustness problem. Also, the C_F^3 generating algorithm can be implemented as an online sampling method that provides guaranteed safe configurations without the *sample-reject-resample* iterations.

8.9 Chapter Summary

This chapter proposes a robot path planning paradigm with explicit parameterization of free space. Two major classes of algorithms are proposed to efficiently solve for narrow passages bottleneck.

An extended *Highway RoadMap* planner is firstly proposed to solve for rigid-body problems. The idea of *C-layers* is presented where each layer represents a fixed orientation of the robot. By using the closed-form Minkwoski

sums between robot parts and obstacles, the collision-free configuration space at each C-layer is parameterized, and pairs of configurations can be safely connected via the *sweep line* process. Configurations with different rotational components can be connected using two local planners. The first one uses the idea of *local C-space*, where each robot part is slightly enlarged, thereby only restricted motions are allowed resulting in a local C-space of each free configuration. A middle configuration can then be searched on the intersection of the convex local C-space of the two configurations in adjacent C-layers. The second local planner generate the *middle C-layer* by sliding a tightly fitted ellipsoid that encloses each body part. Several benchmark schemes with a poly-ellipsoidal robot and ellipsoidal/superquadric obstacles are performed among the Highway RoadMap and probabilistic planners from the Open Motion Planning Library. Different sampling strategies are also applied to the sampling-based planners. The results show that the Highway RoadMap outperforms the multi-query graph-based probabilistic planners on the dense maze maps, both in ellipsoidal and superquadric fitted environment. And it is competitive with single-query planners in searching for a valid path. Real experiments on 2D path planning problems for a humanoid robot being projected to the plane are performed.

To deal with the burden of the curse of dimensionality, a collision-free sample generating algorithm is then proposed via the closed-form Minkowski sums between an ellipsoid and a general convex differentiable surface and an efficient line sweeping process. The proposed algorithm combines random sampling for the shapes of robot and deterministic exploration of the free

space. This hybrid idea unites the sampling-based methods that deal with high dimensionality and the uniformity of discovering free space for a fixed shape of the robot. The generated set of samples, namely C_F^3, are treated as seeds for sampling-based motion planners from OMPL. Benchmark experiments show that the planners using our proposed C_F^3 samples as seeds outperforms those using random sampling methods, especially in the cases when the environment contains narrow corridors and the dimension of the configuration space is high.

Part IV

Conclusion

Chapter 9

Conclusion and Future Work

This dissertation investigated three major properties of Euclidean motions in robotics systems: quantization, calibration and planning. Several novel group-theoretic and geometric approaches were applied as fundamental mathematical tools to solve for these challenges. The dissertation started from quantizing the Euclidean motions using a concept of discrete alphabet and solved for signal-to-symbol problems to describe a motion sequence by representative primitives. The pre-computed motion alphabet was then used to efficiently locate a given specific robot pose via a coarse-to-fine decoding algorithm. After the quantization, a multi-robot sensor calibration problem was solved for the case when the data was lack of temporal correspondence and had large measurement noise. Both these topics were related to sensing and characterizing the planning arena in a robotic task. Then the geometric motion planning problems were studied, where the rigid parts of the robot were modeled as ellipsoids. At first, the collision and containment queries were comprehensively studied. Followed by these two essential subroutines, motion planning frameworks were introduced and verified by benchmarks

and physical experiments. The proposed frameworks were able to deal with both narrow passage and dimensionality challenges in the field of motion planning. The following summarizes the three topics in details and points some potential future work.

9.1 Quantization of Euclidean Motions

Quantization of Euclidean motions was studied via the concept of motion alphabet, where organized lists of motion primitives were computed and stored. The theories of double-coset decomposition for Lie groups and the properties in the cases of $\mathrm{SO}(3)$, $\mathrm{SE}(2)$, and $\mathrm{SE}(3)$ were studied in details. By this means, the group can be divided into equi-volumetric Voronoi-like cells using appropriate metric functions. Then, to quickly locate a specific pose, a coarse-to-fine decoding algorithm was proposed: it first searched for the fundamental domain of the single coset decomposition, then precisely located the appropriate cell, resulting in a two-letter word. Examples in both 2D and 3D were studied and conceptually visualized. Comparisons with existing discretization algorithms were conducted, which showed the high uniformity and fast query speed properties of the proposed quantization framework.

Since computing the double-coset decomposition required a finite discrete subgroup, the resolution of the quantization was fixed, and only a limited number of subgroups were satisfied. Therefore, in order to fulfill the needs for a flexible resolution, which can be arbitrarily changed, it is possible to further generate a grid around the identity of each fundamental domain. This is because that the area around identity can be approximated as an

Euclidean space, which can be evenly divided into grids. Also, to further verify the applicability to real-world problems, some simulated and physical experiments for rigid-body robots (such as mobile robot, aerial vehicles, etc) can be conducted in the future.

9.2 Probabilistic Calibration of Rigid-body Transformations

The study of calibration problem focused on the multi-robot case, i.e. the $AXB = YCZ$ formulation. To deal with the lack of correspondence of data streams, the previously proposed probabilistic framework was applied. The approximations of the data as Gaussian distribution on the group of rigid-body motions were analogous to the foundation of probabilistic calibration framework. To further mitigate the effects of large covariance of noise, a novel algorithm was proposed. The new algorithm used the combined dataset and iteratively minimized the variations of small motions. The iterative algorithm was verified to solve the problem more effectively through both simulations and physical experiments.

The same spirits of iterative refinement can also be applied to the other calibration problems, such as $AX = XB$ and $AX = YB$, which can be further implemented, verified and experimented.

9.3 Geometric Motion Planning using Closed-form Minkowski Sums

The closed-form Minkowski sums between an ellipsoid and a general differentiable surface was derived as an extension to the previously proposed case of two ellipsoids. This geometric characterization inspired several interesting solutions to different aspects of motion planning. At first, collision detection and proximity queries were solved, which used the boundary of closed-form Minkowski sums and optimized the distance with a given point. Then, the kinematics of containment studied the allowable space of motions for an ellipsoid being fully inside another one. Two useful lower bounds that describe such a motion space were computed. Finally, two algorithms that solved for the geometric motion planning problems in narrow passages were proposed and benchmarked with sampling-based planners, showing efficiency and scalability to high dimensional problems.

The Highway Roadmap algorithm developed here could foster interesting extensions to multi-robot planning problems on graph. Also, the closed-form Minkowski sum itself can also be implemented fast enough to accelerate decoupled multi-robot planning, i.e. using the velocity obstacle idea.

Appendix A

Geometric Properties of Superquadrics and Ellipsoids

A.1 Explicit Expressions of Closed-form Minkowski Sums for Superquadrics

From Eq. (3.51), when the pose of superquadric S_a and the orientation of ellipsoid E_b are fixed, the parameters derived from unknown variables are the parametric surface of S_a, i.e. $\boldsymbol{x}$ and its gradient $\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x})$. Therefore, this appendix provides some necessary and explicit calculations of these two parameters that defines the closed-form Minkowski sums between an ellipsoid and a superquadric surface in both 2D and 3D cases.

A.1.1 The 2D case

The implicit and explicit equations for a superellipse S_a in $\mathbb{R}^2$ are defined as

$$\begin{aligned} \Phi(x_1, y_1) &= \left(\frac{x_1}{a_1}\right)^{\frac{2}{\epsilon}} + \left(\frac{y_1}{b_1}\right)^{\frac{2}{\epsilon}} = 1\,, \text{ and} \\ \boldsymbol{x} &= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} a_1 \cos^{\epsilon}\theta \\ b_1 \sin^{\epsilon}\theta \end{bmatrix}, \end{aligned} \tag{A.1}$$

where $-\pi \leq \theta \leq \pi$, respectively. The shape described by the above function changes with ϵ. Only the case of $0 < \epsilon < 2$ is considered to ensure the convexity of the corresponding shape. The gradient of $\Phi(x_1(\theta), y_1(\theta))$ with respect to the parameter θ can be computed as

$$\nabla\Phi(x_1(\theta), y_1(\theta)) = \frac{2}{\epsilon}\begin{bmatrix} \cos^{2-\epsilon}\theta / a_1 \\ \sin^{2-\epsilon}\theta / b_1 \end{bmatrix}. \tag{A.2}$$

A.1.2 The 3D case

The implicit and explicit equations for a superquadric surface S_a in $\mathbb{R}^3$ are defined as

$$\begin{aligned} \Phi(x_1, y_1, z_1) &= \left(\left(\frac{x_1}{a_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y_1}{b_1}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z_1}{c_1}\right)^{\frac{2}{\epsilon_1}} = 1\,, \text{ and} \\ \boldsymbol{x} &= \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} a_1 \cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega \\ b_1 \cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega \\ c_1 \sin^{\epsilon_1}\eta \end{bmatrix}, \end{aligned} \tag{A.3}$$

where $-\pi/2 \leq \eta \leq \pi/2,\ -\pi \leq \omega \leq \pi$, respectively. The surface described by the above function changes with ϵ_1 and ϵ_2, and only the case of $0 < \epsilon_1, \epsilon_2 < 2$ is considered to ensure the convexity of the superquadric surface.

The gradient of $\Phi(x_1(\eta,\omega), y_1(\eta,\omega), z_1(\eta,\omega))$ with respect to the parameters η and ω can be computed as

$$\nabla\Phi(x_1(\eta,\omega), y_1(\eta,\omega), z_1(\eta,\omega)) = \frac{2}{\epsilon_1}\begin{bmatrix}\cos^{2-\epsilon_1}\eta\cos 2 - \epsilon_2\omega/a_1\\ \cos^{2-\epsilon_1}\eta\sin 2 - \epsilon_2\omega/b_1\\ \sin^{2-\epsilon_1}\eta/c_1\end{bmatrix}. \quad \text{(A.4)}$$

A.2 Superquadric Reconstructions from Point Cloud Data

Comparing to mesh-based reconstruction of the surface, using superquadrics to encapsulate a rigid body is equivalent convenient. The procedure of recovering the corrsponding superquadric model from point cloud data is reviewed here.

A.2.1 The 3D case

Given a set of m 3D points on the surface of an object, i.e. $\{(x_i, y_i, z_i),\ i = 1, ..., m\}$, the objective is to to minimize the total distance with the superquadric surface while keeping all the points inside. The inside-outside implicit equation of a superquadric model is given by

$$F(x,y,z) = \left(\left(\frac{x}{a}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{c}\right)^{\frac{2}{\epsilon_1}} = 1,$$

where a, b, c are the semi-axes lengths, and ϵ_1, ϵ_2 are the exponents that determine the roundness of the surface. To fit the point data into the model, each point is firstly transformed into the body frame of the superquadric by

$g \doteq (R, \boldsymbol{t}) \in \mathrm{SE}(3)$ as

$$[x_i', y_i', z_i']^\top = R^\top([x_i, y_i, z_i]^\top - \boldsymbol{t}),$$

and construct the objective function $\|F^{\epsilon_1}(x_i', y_i', z_i') - 1\|$ that is related to the radial distance of the point from the surface. And to force the resulting surface enclose every point, a hard constraint is applied as $F(x_i', y_i', z_i') \leq 1$. Therefore, the nonlinear optimization problem can be formulated as [87]

$$\begin{aligned} \min_{a,b,c,\epsilon_1,\epsilon_2,R,\boldsymbol{t}} \; & abc \sum_{i=1}^{m} \left(F^{\epsilon_1}(x_i', y_i', z_i') - 1\right)^2, \\ \text{s.t. } & F(x_i', y_i', z_i') \leq 1. \end{aligned} \tag{A.5}$$

Note that (x_i', y_i', z_i') is the transformed data point as viewed in the body frame of the superquadric, and the factor abc is added here in order to impose the smallest superquadric that fits the point cloud data (see [137] for more details).

A.2.2 The 2D case

The implicit expression for a 2D superellipse is

$$G(x, y) = \left(\frac{x}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon}} = 1,$$

where a, b are the semi-axes lengths, and ϵ is the exponent that determines the roundness of the boundary curve. Then the optimization to fit a set of m 2D points on the boundary curve of a 2D object transformed in the body frame of

the object, i.e. $\{(x'_i, y'_i),\ i = 1, ..., m\}$ is formulated as

$$\begin{aligned} &\min_{a,b,\epsilon,\theta,\boldsymbol{t}} ab \sum_{i=1}^{m} \left(G^{\epsilon}(x'_i, y'_i) - 1\right)^2, \\ &\text{s.t. } G(x'_i, y'_i) \leq 1. \end{aligned} \tag{A.6}$$

In this case, there are 6 variables to be optimized, where θ and $\boldsymbol{t} \in \mathbb{R}^2$ are the rotational angle and coordinates of the center of the superellipse respectively.

A.2.3 The initial guess

Furthermore, an initial guess is necessary for solving the problem. Rather than randomly select initial variables, an ellipsoidal model is chosen to fit the data points at first. In particular, the semi-axes are computed from the principle components of the data. This model captures both the center and the directions of the largest variances of the data points, which can be a good starting point to solve the optimization problem.

A.3 Proof of the Volume Minimality of the Concentric Ellipsoid Containing Two Ellipsoids

This section proves the volume minimality of the concentric ellipsoid E_m that contains both E_a and E_b via Eq. (3.54).

In the shrunk space, it is clear that E'_m reaches the minimum volume when its semi-axes are aligned with those of E'_a because of the symmetry of $E'_a \cup E'_b$. Then, the implicit expressions of E'_m can be written as

$$\Phi_{E'_m}(\mathbf{x}) = \mathbf{x}^\top R'_a \Lambda^{-2}(\mathbf{m}') R'^\top_a \mathbf{x}, \tag{A.7}$$

because of the alignment of the semi-axes. Now a change of variables is applied as $\mathbf{y} = R_a'^{\top}\mathbf{x}$, then Eq. (A.7) becomes

$$\Phi_{E_m'}(\mathbf{y}) = \mathbf{y}^{\top}\Lambda^{-2}(\mathbf{m}')\mathbf{y} = \sum_{i=1}^{n}\frac{y_i^2}{m_i'^2}. \tag{A.8}$$

The implicit expressions for E_a' and E_b', after the change of variables, are

$$\Phi_{E_a'}(\mathbf{y}) = \sum_{i=1}^{n}\frac{y_i^2}{a_i'^2} \quad \text{and} \quad \Phi_{E_b'}(\mathbf{y}) = \sum_{i=1}^{n}\frac{y_i^2}{r^2}, \tag{A.9}$$

respectively.

Firstly, when $m_i' = \max(a_i', r)$ $(i = 1, ..., n)$, E_m' contains both E_a' and E_b'. Suppose an arbitrary point $\mathbf{y}_0$ lies inside or on the boundary of E_a', then $\Phi_{E_a'}(\mathbf{y}_0) = \sum_{i=1}^{n}\frac{y_i^2}{a_i'^2} \leq 1$. Then, since $a_i' \leq \max(a_i', r) = m_i'$, then $\Phi_{E_m'}(\mathbf{y}_0) = \sum_{i=1}^{n}\frac{y_i^2}{m_i'^2} \leq \sum_{i=1}^{n}\frac{y_i^2}{a_i'^2} \leq 1$, which means that the point $\mathbf{y}_0$ is also inside or on the boundary of E_m'. The same procedure can be apply when $\mathbf{y}_0$ is contained in E_b'.

Next, by contradiction, m_i' can be shown to be greater than or equal to $\max(a_i', r)$ $(i = 1, ..., n)$ to ensure the containment as follows. Assume that an arbitrary k^{th} semi-axis of E_m' satisfies $m_k' < \max(a_k', r)$. Now, without loss of generality, suppose $a_k' \geq r$, then $m_k' < \max(a_k', r) = a_k'$. And if $\mathbf{y}_k = a_k'\mathbf{e}_k$ ($\mathbf{e}_k$ is the k^{th} basis vector of $\mathbb{R}^n$) is on the boundary and the k^{th} semi-axis of E_a', then $\Phi_{E_a'}(\mathbf{y}_k) = \frac{a_k'^2}{a_k'^2} = 1$. Substituting $\mathbf{y}_k$ into Eq. (A.7) gives $\Phi_{E_m'}(\mathbf{y}_k) = \frac{a_k'^2}{m_k'^2} > 1$, which implies that $\mathbf{y}_k$ is outside E_m', and therefore contradicts that E_m' contains both E_a' and E_b'.

Combining the above statements, it can be concluded that the minimum values of the semi-axes lengths that make E_m' containing both E_a' and E_b' is

$m_i'^* = \max(a_i', r)$ $(i = 1, ..., n)$. The volume expressions for n-dimensional ellipsoids give

$$Vol^*(E_m') = \eta \prod_{i=1}^{n} \max(a_i', r), \tag{A.10}$$

where η is the volume of a unit sphere in $\mathbb{R}^n$. From the property of affine transformation, the corresponding volume of E_m, with the equal sign holds, in the original space is

$$\begin{aligned} Vol(E_m^*) &= \eta |\det(T)| \prod_{i=1}^{n} \max(a_i', r) \\ &= \frac{\eta}{r^n} \prod_{i=1}^{n} b_i \max(a_i', r). \end{aligned} \tag{A.11}$$

Since $\det(T)$ is constant, the minimality is preserved, thus $Vol(E_m^*)$ is minimal, which concludes the proof.

Note that, no matter which ellipsoid to be shrunk into a sphere, the results will be the same: even though the shrunk spaces are generated by different affine transformations, i.e. T_a and T_b, the covering ellipsoids E_{ma}' and E_{mb}' always have the minimum volume in the corresponding space. Therefore, after the inverse affine transformation, the resulting E_{ma} and E_{mb} both have minimum volume in the original space, which indicates the uniqueness of the result, i.e. $Vol(E_m) = Vol(E_{ma}) = Vol(E_{mb})$.

A.4 Computations of the Extreme Distance a Sphere Can Move Along the Semi-axis of an Ellipsoid in $\mathbb{R}^n$

Without loss of generality, the range of the extreme distance is limited to be $d_i^* \in [0, b_i' - r]$. The upper bound, which is the largest distance that E_a' can move along each semi-axis. Expanding Eq. (7.10c) and defining $B' \doteq \Lambda^{-2}(\boldsymbol{b}')$ gives

$$(B - k\mathbb{I})\boldsymbol{x}_0 = -kd\boldsymbol{e}_i. \tag{A.12}$$

Since $B - k\mathbb{I}$ is diagonal, Eq. (A.12) holds in just two cases:

(I) $\boldsymbol{x}_0 = x_i\boldsymbol{e}_i$ ($x_i \neq 0$); or

(II) $\boldsymbol{x}_0 = x_i\boldsymbol{e}_i + \sum_j x_j\boldsymbol{e}_j$, where $\forall j, k = b_j'^{-2}$ ($i \neq j$, $x_i, x_j \neq 0$).

Geometrically, case (I) or (II) happens when the vector from origin to the touching point is or is not parallel to the i^{th} semi-axis respectively. As a result, those two cases cover all possible situations of the solution.

Case (I): when $\boldsymbol{x}_0 = x_i\boldsymbol{e}_i$ ($x_i \neq 0$), Eq. (7.10a) and (7.10b) become $x_i^2 b_i'^{-2} = 1$ and $d^2 - 2x_i d + x_i^2 - r^2 = 0$ respectively. Solving for d gives $d = x_i \pm r = \pm b_i' \pm r$. Combining with the range of the extreme distance gives

$$d_i^* = b_i' - r. \tag{A.13}$$

Case (II): Substituting $\boldsymbol{x}_0$ into Eq. (7.10a) and (7.10b) gives

$$\sum_j (x_j^2 b_j^{'-2}) + x_i^2 b_i^{'-2} = 1, \tag{A.14a}$$

$$\sum_j (x_j^2) + (x_i - d)^2 = r^2. \tag{A.14b}$$

Since $\forall j, b_j^{'-2} = k$ are the same, it can be grouped out from the summation in Eq. (7.10a) as

$$b_j^{'-2} \sum_j (x_j^2) + x_i^2 b_i^{'-2} = 1. \tag{A.15}$$

Further, substituting the equation of the sphere into that of the ellipsoid gives

$$b_j^{'-2}[r^2 - (x_i - d)^2] + x_i^2 b_i^{'-2} = 1. \tag{A.16}$$

Using the condition $k = b_j^{'-2}$, Eq. (A.12) becomes

$$(b_i^{'-2} - b_j^{'-2}) x_i = -b_j^{'-2} d, \tag{A.17}$$

where x_i and b_i' are fixed since the touching points are fixed once the shapes of the ellipsoid and sphere are given. The index j can be searched when b_{j^*}' is the maximum of all the semi-axes other than b_i', because the extreme distance should be the minimum among all possible values. Therefore,

$$j^* = \arg\max_{j \neq i} (b_j'). \tag{A.18}$$

Combining Eq. (A.16), (A.17) and (A.18) and solving for d gives

$$d_i^* = \frac{1}{b_{j^*}'} \sqrt{\left(b_{j^*}'^2 - b_i'^2\right)\left(r^2 - b_{j^*}'^2\right)}. \tag{A.19}$$

The critical value of the radius r between the two cases can be calculated

by equating (A.13) and (A.19) as

$$r = b'^2_{j^*}/b'_i. \tag{A.20}$$

And from the geometric point of view, when r is less than this critical value, the sphere should touch the end point of the ellipsoid at the i^{th} semi-axis, which is in case (I); otherwise, the extreme distance according to case (II) should be computed.

Appendix B

Derivations of Probabilities on $\mathbf{SE}(3)$ for the $AXB = YCZ$ Calibration Problem

B.1 Second-order Approximation of the Three-fold Convolution of Probability Density Functions

The associativity of the group operations give

$$((f_1 * f_2) * f_3)(g) \;=\; (f_1 * (f_2 * f_3))(g) \;=\; (f_1 * f_2 * f_3)(g) \tag{B.1}$$

and

$$\Sigma_{(1*2)*3} = \Sigma_{1*(2*3)} = \Sigma_{1*2*3}\,. \tag{B.2}$$

Let us now evaluate the three-fold convolution using both of the above equalities and the properties of $F(\cdot\,;\,\cdot)$ in Eq. (3.39):

$$\Sigma_{(1*2)*3} = Ad(M_3^{-1})\Sigma_{1*2}Ad^T(M_3^{-1}) + \Sigma_3 + F\left(Ad(M_3^{-1})\Sigma_{1*2}Ad^T(M_3^{-1})\,;\,\Sigma_3\right), \tag{B.3}$$

where $\Sigma_{1*2} = Ad(M_2^{-1})\Sigma_1 Ad^T(M_2^{-1}) + \Sigma_2 + F\left(Ad(M_2^{-1})\Sigma_1 Ad^T(M_2^{-1})\,;\,\Sigma_2\right)$.

If $\Sigma_1 = \Sigma_2 = \mathbb{O}$,

$$\Sigma_{1*2*3} = \Sigma_3; \tag{B.4}$$

and if $\Sigma_2 = \Sigma_3 = \mathbb{O}$,

$$\Sigma_{1*2*3} = Ad_{M_3}^{-1} Ad_{M_2}^{-1} \Sigma_1 Ad_{M_2}^{-T} Ad_{M_3}^{-T}. \tag{B.5}$$

Hence both of the covariances do not depend on the second-order term.

B.2 Explicit Solutions to the Iterative Algorithm for Probabilistic $AXB = YCZ$ Calibration Problem

The equations to be solved are

$$A_i X M_{B_i} = Y M_{C_i} Z \tag{B.6a}$$

$$\Sigma_{B_i} = Ad(Z^{-1}) \Sigma_{C_i} Ad^{\top}(Z^{-1}) \tag{B.6b}$$

$$C_j Z M_{B_j}^{-1} = Y^{-1} M_{A_j} X \tag{B.6c}$$

$$\Sigma_{B_j^{-1}} = Ad(X^{-1}) \Sigma_{A_j} Ad^{\top}(X^{-1}) \tag{B.6d}$$

for i, j are the number of trials of dataset for fixing A_i and C_j respectively.

A small perturbation of the initial X, Y and Z can be expressed as

$$\begin{aligned} X_{k+1} &= X_k(\mathbb{I} + \widehat{\boldsymbol{\xi}}_{X_k}) \\ Y_{k+1} &= Y_k(\mathbb{I} + \widehat{\boldsymbol{\xi}}_{Y_k}) \\ Z_{k+1} &= Z_k(\mathbb{I} + \widehat{\boldsymbol{\xi}}_{Z_k}) \end{aligned} \tag{B.7}$$

where, $\boldsymbol{\xi}_k = [\boldsymbol{\omega}_k, \boldsymbol{v}_k]^\top$.

The following gives a detailed derivation of the explicit forms of each part of the matrices P_k and $\boldsymbol{b}_k$

B.2.1 Construction of P_{1k} and $\mathbf{b}_{1k}$

Substituting Eq. (B.7) back into Eq. (B.6a) and eliminating quadratic terms gives

$$A_i X_k \widehat{\boldsymbol{\xi}}_{X_k} M_{B_i} - Y_k \widehat{\boldsymbol{\xi}}_{Y_k} M_{C_i} Z_k - Y_k M_{C_i} Z_k \widehat{\boldsymbol{\xi}}_{Z_k} = -A_i X_k M_{B_i} + Y_k M_{C_i} Z_k. \tag{B.8}$$

Separating the rotation and translation parts gives,

(1) Rotation part:

$$\begin{aligned} & R_{A_i} R_{X_k} \widehat{\boldsymbol{\omega}}_X R_{M_{B_i}} - R_{Y_k} R_{M_{C_i}} R_{Z_k} \widehat{\boldsymbol{\omega}}_Z - R_{Y_k} \widehat{\boldsymbol{\omega}}_Y R_{M_{C_i}} R_{Z_k} \\ = & - R_{A_i} R_{X_k} R_{M_{B_i}} + R_{Y_k} R_{M_{C_i}} R_{Z_k} \,. \end{aligned} \tag{B.9}$$

Using the fact that $\boldsymbol{a} \times \boldsymbol{b} = \widehat{\boldsymbol{a}}\boldsymbol{b} = -\widehat{\boldsymbol{b}}\boldsymbol{a}$ and treating each column separately

gives

$$
\begin{aligned}
&- R_{A_i} R_{X_k} \widehat{(R_{M_{B_i}})}_m \boldsymbol{\omega}_{X_k} + R_{Y_k} R_{M_{C_i}} R_{Z_k} \widehat{\boldsymbol{e}}_m \boldsymbol{\omega}_{Z_k} + R_{Y_k} \widehat{(R_{M_{C_i}} R_{Z_k})}_m \boldsymbol{\omega}_{Y_k} \\
=&Rot(-A_i X_k M_{B_i} + Y_k M_{C_i} Z_k)_m
\end{aligned}
\tag{B.10}
$$

where $m = 1, 2, 3$ is the index of columns of the rotation matrices, and $Rot(\cdot)$ is the rotation part of a rigid body transformation matrix.

(2) Translation part:

$$
\begin{aligned}
&- R_{A_i} R_{X_k} \widehat{\boldsymbol{t}}_{M_{B_i}} \boldsymbol{\omega}_{X_k} + R_{Y_k} (R_{M_{C_i}} \widehat{\boldsymbol{t}_{Z_k}} + \boldsymbol{t}_{M_{C_i}}) \boldsymbol{\omega}_{Y_k} + R_{A_i} R_{X_k} \boldsymbol{v}_{X_k} \\
&- R_{Y_k} \boldsymbol{v}_{Y_k} - R_{Y_k} R_{M_{C_i}} R_{Z_k} \boldsymbol{v}_{Z_k} \\
&= Trans(-A_i X_k M_{B_i} + Y_k M_{C_i} Z_k)
\end{aligned}
\tag{B.11}
$$

where $Trans(\cdot)$ is the translation part of a rigid body transformation matrix.

Combining the rotation and translation parts and stacking $\boldsymbol{\omega}_k$ and $\boldsymbol{v}_k$ together to form a linear system of equations, the corresponding matrices P_{1k} and $\boldsymbol{b}_{1k}$ can be obtained as

$$
P_{1k} = \begin{bmatrix}
P_{1m1} & 0_{3\times3} & P_{1m3} & 0_{3\times3} & P_{1m5} & 0_{3\times3} \\
-R_{A_i} R_{X_k} \widehat{\boldsymbol{t}}_{M_{B_i}} & R_{A_i} R_{X_k} & R_{Y_k} (R_{M_{C_i}} \widehat{\boldsymbol{t}_{Z_k}} + \boldsymbol{t}_{M_{C_i}}) & -R_{Y_k} & 0_{3\times1} & -R_{Y_k} R_{M_{C_i}} R_{Z_k}
\end{bmatrix}
\tag{B.12}
$$

where,

$P_{1m1} = -R_{A_i} R_{X_k} \widehat{(R_{M_{B_i}})}_m$, $P_{1m3} = R_{Y_k} \widehat{(R_{M_{C_i}} R_{Z_k})}_m$, $P_{1m5} = R_{Y_k} R_{M_{C_i}} R_{Z_k} \widehat{\boldsymbol{e}}_m$.

And

$$
\boldsymbol{b}_{1k} = \begin{bmatrix} Rot(-A_i X_k M_{B_i} + Y_k M_{C_i} Z_k)_m \\ Trans(-A_i X_k M_{B_i} + Y_k M_{C_i} Z_k) \end{bmatrix}. \tag{B.13}
$$

B.2.2 Construction of P_{2k} and $\mathbf{b}_{2k}$

Using the similar idea for Eq. (B.6b) gives

$$\Sigma_{B_i} = Ad(Z_{k+1}^{-1})\Sigma_{C_i}Ad^{\top}(Z_{k+1}^{-1}) \tag{B.14}$$

Using the identities of adjoint in Eq. (3.26) and (3.28) gives

$$\begin{aligned} &\Sigma_{B_i} + ad(\widehat{\boldsymbol{\xi}}_Z)\Sigma_{B_i} + \Sigma_{B_i}ad^{\top}(\widehat{\boldsymbol{\xi}}_Z) \\ &= Ad^{-1}(Z_k)\Sigma_{C_i}Ad^{-\top}(Z_k) \end{aligned} \tag{B.15}$$

Splitting the covariance and adjoint matrices into 4 blocks and applying matrix multiplications give,

$$\begin{aligned} &\begin{bmatrix} \hat{\omega}_Z & \mathbb{O} \\ \hat{v}_Z & \hat{\omega}_Z \end{bmatrix} \begin{bmatrix} \Sigma_{B_i}^1 & \Sigma_{B_i}^2 \\ \Sigma_{B_i}^3 & \Sigma_{B_i}^4 \end{bmatrix} + \begin{bmatrix} \Sigma_{B_i}^1 & \Sigma_{B_i}^2 \\ \Sigma_{B_i}^3 & \Sigma_{B_i}^4 \end{bmatrix} \begin{bmatrix} \hat{\omega}_Z^{\top} & \hat{v}_Z^{\top} \\ \mathbb{O} & \hat{\omega}_Z^{\top} \end{bmatrix} \\ &= Ad^{-1}(Z_k)\Sigma_{C_i}Ad^{-\top}(Z_k) - \Sigma_{B_i} \end{aligned} \tag{B.16}$$

Applying block matrix multiplications and for each block, $\boldsymbol{\omega_Z}$ and $\boldsymbol{v}_Z$ can be extracted by treating the columns separately as

$$\begin{aligned} \left[-(\widehat{\Sigma_{B_i}^1})_m + \Sigma_{B_i}^1\widehat{\boldsymbol{e}}_m\right]\omega_Z &= (\mathrm{RHS})_m^1 \\ \left[-(\widehat{\Sigma_{B_i}^2})_m + \Sigma_{B_i}^2\widehat{\boldsymbol{e}}_m\right]\omega_Z + \Sigma_{B_i}^1\widehat{e}_m\boldsymbol{v}_Z &= (\mathrm{RHS})_m^2 \\ \left[-(\widehat{\Sigma_{B_i}^3})_m + \Sigma_{B_i}^3\widehat{\boldsymbol{e}}_m\right]\omega_Z - (\widehat{\Sigma_{B_i}^1})_m\boldsymbol{v}_Z &= (\mathrm{RHS})_m^3 \\ \left[-(\widehat{\Sigma_{B_i}^4})_m + \Sigma_{B_i}^4\widehat{\boldsymbol{e}}_m\right]\omega_Z + \left[-(\widehat{\Sigma_{B_i}^2})_m + \Sigma_{B_i}^3\widehat{e}_m\right]\boldsymbol{v}_Z &= (\mathrm{RHS})_m^4 \end{aligned} \tag{B.17}$$

where $\mathrm{RHS} = Ad^{-1}(Z_k)\Sigma_{C_i}Ad^{-\top}(Z_k) - \Sigma_{B_i}$, $m = 1, 2, 3$ denotes the index of the column on each matrix or block of matrix, and the right superscripts

denotes the index of the block in the matrix.

In this case, the P_{2k} and $\boldsymbol{b}_{2k}$ matrices are

$$P_{2k} = \begin{bmatrix} 0_{3\times 12} & -(\widehat{\Sigma^1_{B_i}})_m + \Sigma^1_{B_i}\widehat{\boldsymbol{e}}_m & 0_{3\times 3} \\ 0_{3\times 12} & -(\widehat{\Sigma^2_{B_i}})_m + \Sigma^2_{B_i}\widehat{\boldsymbol{e}}_m & \Sigma^1_{B_i}\widehat{e}_m \\ 0_{3\times 12} & -(\widehat{\Sigma^3_{B_i}})_m + \Sigma^3_{B_i}\widehat{\boldsymbol{e}}_m & -(\widehat{\Sigma^1_{B_i}})_m \\ 0_{3\times 12} & -(\widehat{\Sigma^4_{B_i}})_m + \Sigma^4_{B_i}\widehat{\boldsymbol{e}}_m & -(\widehat{\Sigma^2_{B_i}})_m + \Sigma^3_{B_i}\widehat{\boldsymbol{e}}_m \end{bmatrix}, \tag{B.18}$$

and

$$\boldsymbol{b}_{2k} = \begin{bmatrix} (Ad^{-1}(Z_k)\Sigma_{C_i}Ad^{-\top}(Z_k) - \Sigma_{B_i})^1_m \\ (Ad^{-1}(Z_k)\Sigma_{C_i}Ad^{-\top}(Z_k) - \Sigma_{B_i})^2_m \\ (Ad^{-1}(Z_k)\Sigma_{C_i}Ad^{-\top}(Z_k) - \Sigma_{B_i})^3_m \\ (Ad^{-1}(Z_k)\Sigma_{C_i}Ad^{-\top}(Z_k) - \Sigma_{B_i})^4_m \end{bmatrix}. \tag{B.19}$$

B.2.3 Construction of P_{3k} and $\mathbf{b}_{3k}$

The matrix P_{3k} and b_{3k} can be obtained explicitly, using the same ideas above, as

$$P_{3k} = \begin{bmatrix} R_{Y_k^{-1}}R_{M_{A_j}}R_{X_k}\widehat{\boldsymbol{e}}_m & 0_{3\times 3} & -(\widehat{R_{Y_k^{-1}}R_{M_{A_j}}R_{X_k}})_m & 0_{3\times 3} & -R_{C_j}R_{Z_k}(\widehat{R_{M_{B_j}^{-1}}})_m & 0_{3\times 3} \\ 0_{3\times 3} & P_{342} & P_{343} & \mathbb{I}_{3\times 3} & P_{345} & P_{346} \end{bmatrix} \tag{B.20}$$

where,

$P_{342} = -R_{Y_k^{-1}}R_{M_{A_j}}R_{X_k}$, $P_{343} = -(\widehat{R_{Y_k^{-1}}R_{M_{A_j}}t_{X_k} + R_{Y_k^{-1}}t_{M_{A_j}}} + t_{Y_k^{-1}})$, $P_{345} = -R_{C_j}R_{Z_k}\widehat{t_{M_{B_j}^{-1}}}$, $P_{346} = R_{C_j}R_{Z_k}$.

And

$$\boldsymbol{b}_{3k} = \begin{bmatrix} Rot(-C_jZ_kM_{B_j}^{-1} + Y_k^{-1}M_{A_j}X_k)_m \\ Trans(-C_jZ_kM_{B_j}^{-1} + Y_k^{-1}M_{A_j}X_k) \end{bmatrix} \tag{B.21}$$

B.2.4 Construction of P_{4k} and $\mathbf{b}_{4k}$

Eq. (B.6d) shows that

$$\Sigma_{B_j^{-1}} = Ad(X_{k+1}^{-1})\Sigma_{A_j}Ad^{\top}(X_{k+1}^{-1}) \tag{B.22}$$

Note that when actually calculating the inverse of Σ_B, it is better to use $\Sigma_{B_j^{-1}} = Ad(B)\Sigma_B Ad^{\top}(B)$.

Using the same methodology with previous derivations, the explicit forms of P_{4k} and $\boldsymbol{b}_{4k}$ can be written as

$$P_{4k} = \begin{bmatrix} -(\widehat{(\Sigma_{B_j^{-1}})^1})_m + (\Sigma_{B_j^{-1}})^1\widehat{\boldsymbol{e}}_m & 0_{3\times 3} & 0_{3\times 12} \\ -(\widehat{(\Sigma_{B_j^{-1}})^2})_m + (\Sigma_{B_j^{-1}})^2\widehat{\boldsymbol{e}}_m & (\Sigma_{B_j^{-1}})^1\widehat{\boldsymbol{e}}_m & 0_{3\times 12} \\ -(\widehat{(\Sigma_{B_j^{-1}})^3})_m + (\Sigma_{B_j^{-1}})^3\widehat{\boldsymbol{e}}_m & -(\widehat{(\Sigma_{B_j^{-1}})^1})_m & 0_{3\times 12} \\ -(\widehat{(\Sigma_{B_j^{-1}})^4})_m + (\Sigma_{B_j^{-1}})^4\widehat{\boldsymbol{e}}_m & -(\widehat{(\Sigma_{B_j^{-1}})^2})_m + (\Sigma_{B_j^{-1}})^3\widehat{\boldsymbol{e}}_m & 0_{3\times 12} \end{bmatrix}, \tag{B.23}$$

and

$$\boldsymbol{b}_{4k} = \begin{bmatrix} (Ad^{-1}(X_k)\Sigma_{A_j}Ad^{-\top}(X_k) - \Sigma_{B_j^{-1}})_m^1 \\ (Ad^{-1}(X_k)\Sigma_{A_j}Ad^{-\top}(X_k) - \Sigma_{B_j^{-1}})_m^2 \\ (Ad^{-1}(X_k)\Sigma_{A_j}Ad^{-\top}(X_k) - \Sigma_{B_j^{-1}})_m^3 \\ (Ad^{-1}(X_k)\Sigma_{A_j}Ad^{-\top}(X_k) - \Sigma_{B_j^{-1}})_m^4 \end{bmatrix}. \tag{B.24}$$

In the end, P_k and $\boldsymbol{b}_k$ can be obtained by concatenating the four parts calculated above.

Bibliography

[1] M. K. Ackerman and G. S. Chirikjian, "A probabilistic solution to the AX = XB problem: Sensor calibration without correspondence," in *Geometric Science of Information*, Springer, 2013, pp. 693–701.

[2] E. I. Agba, T.-L. Wong, M. Z. Huang, and A. M. Clark, "Objects interaction using superquadrics for telemanipulation system simulation," *Journal of robotic systems*, vol. 10, no. 1, pp. 1–22, 1993.

[3] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 155–168.

[4] M. A. Arbib, "Theories of abstract automata," *Journal of Symbolic Logic*, vol. 37, no. 2, pp. 412–413, 1972.

[5] M. Aroyo, J. Perez-Mato, C. Capillas, E. Kroumova, S. Ivantchev, G. Madariaga, A. Kirov, and H. Wondratschek, "Bilbao crystallographic server: I. databases and crystallographic computing programs," *Zeitschrift für Kristallographie*, vol. 221, no. 1, pp. 15–27, 2006.

[6] J. Arvo, "Fast random rotation matrices," in *Graphics Gems III (IBM Version)*, Elsevier, 1992, pp. 117–120.

[7] J.-P. Aubin and H. Frankowska, *Set-valued analysis*. Springer Science & Business Media, 2009.

[8] A. Badawy and C. McInnes, "Separation distance for robot motion control using superquadric obstacle potentials," in *International Control Conference*, 2006.

[9] C. Bajaj, A. Bhowmick, E. Chattopadhyay, and D. Zuckerman, "On low discrepancy samplings in product spaces of motion groups," *arXiv preprint arXiv:1411.7753*, 2014.

[10] H. Baker, "Alternants and continuous groups," *Proceedings of the London Mathematical Society*, vol. 2, no. 1, pp. 24–47, 1905.

[11] H. Barki, F. Denis, and F. Dupont, "Contributing vertices-based Minkowski sum computation of convex polyhedra," *Computer-Aided Design*, vol. 41, no. 7, pp. 525–538, 2009.

[12] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer graphics and Applications*, vol. 1, no. 1, pp. 11–23, 1981.

[13] E. Behar and J.-M. Lien, "Dynamic Minkowski sum of convex shapes," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2011, pp. 3463–3468.

[14] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational geometry: algorithms and applications*. Springer-Verlag Telos, 2008, pp. 290–292.

[15] D. P. Bertsekas, *Convex optimization theory*. Athena Scientific Belmont, 2009.

[16] A. Best, S. Narang, and D. Manocha, "Real-time reciprocal collision avoidance with elliptical agents," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 298–305.

[17] G. Birkhoff and S. Lane, *A Survey of Modern Algebra*. Taylor & Francis, 1998.

[18] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *IEEE International Conferencce on Robotics and Automation (ICRA)*, IEEE, vol. 1, 2000, pp. 521–528.

[19] V. Boor, M. H. Overmars, and A. F. Van Der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *IEEE International Conferencce on Robotics and Automation (ICRA)*, 1999, pp. 1018–1023.

[20] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.

[21] N. Chakraborty, J. Peng, S. Akella, and J. E. Mitchell, "Proximity queries between convex objects: An interior point approach for implicit surfaces," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 211–220, 2008.

[22] B. Chazelle, "Approximation and decomposition of shapes," *Algorithmic and Geometric Aspects of Robotics*, vol. 1, pp. 145–185, 1985.

[23] ——, *The discrepancy method: randomness and complexity*. Cambridge University Press, 2001.

[24] L. Chen, C. Ni, J. Feng, J. Dai, B. Huang, H. Liu, and H. Pan, "Proximity query based on second order cone programming using convex superquadrics: A static collision detection algorithm for narrow-phase," *Assembly Automation*, vol. 35, no. 4, pp. 367–375, 2015.

[25] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011, vol. 2.

[26] G. S. Chirikjian, "Kinematics meets crystallography: The concept of a motion space," *Journal of Computing and Information Science in Engineering*, vol. 15, no. 1, 2015.

[27] ——, "Mathematical aspects of molecular replacement. I. algebraic properties of motion spaces," *Acta Crystallographica Section A: Foundations of Crystallography*, vol. 67, no. 5, pp. 435–446, 2011.

[28] ——, "Parts entropy and the principal kinematic formula," in *CASE*, IEEE, 2008, pp. 864–869.

[29] ——, "Parts entropy and the principal kinematic formula," in *Stochastic Models, Information Theory, and Lie Groups, Volume 2*, Springer, 2012, pp. 187–228.

[30] G. S. Chirikjian and A. B. Kyatkin, *Harmonic Analysis for Engineers and Applied Scientists: Updated and Expanded Edition*. Courier Dover Publications, 2016.

[31] G. S. Chirikjian, R. Mahony, S. Ruan, and J. Trumpf, "Pose changes from a different point of view," *Journal of Mechanisms and Robotics*, vol. 10, no. 2, p. 021 008, 2018.

[32] G. S. Chirikjian, K. Ratnayake, and S. Sajjadi, "Decomposition of Sohncke space groups into products of Bieberbach and symmorphic parts," *Zeitschrift für Kristallographie-Crystalline Materials*, vol. 230, no. 12, pp. 719–741, 2015.

[33] G. S. Chirikjian, S. Sajjadi, B. Shiffman, and S. M. Zucker, "Mathematical aspects of molecular replacement. IV. measure-theoretic decompositions of motion spaces," *Acta Crystallographica Section A: Foundations and Advances*, vol. 73, no. 5, pp. 387–402, 2017.

[34] G. S. Chirikjian and Y. Yan, "Mathematical aspects of molecular replacement. II. geometry of motion spaces," *Acta Crystallographica Section A: Foundations of Crystallography*, vol. 68, no. 2, pp. 208–221, 2012.

[35] ——, "The kinematics of containment," in *Advances in Robot Kinematics*, Springer, 2014, pp. 355–364.

[36] Y.-K. Choi, J.-W. Chang, W. Wang, M.-S. Kim, and G. Elber, "Continuous collision detection for ellipsoids," *IEEE Transactions on visualization and Computer Graphics*, vol. 15, no. 2, pp. 311–325, 2009.

[37] Y.-K. Choi, W. Wang, and M.-S. Kim, "Exact collision detection of two moving ellipsoids under rational motions," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2003, pp. 349–354.

[38] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[39] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scale environments," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, ACM, 1995, 189–ff.

[40] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*. Springer, 2017, vol. 118.

[41] F. H. Crick, L. Barnett, S. Brenner, and R. J. Watts-Tobin, "General nature of the genetic code for proteins," *Nature*, vol. 192, no. 4809, pp. 1227–1232, 1961.

[42] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[43] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, "Computational geometry," in *Computational geometry*, Springer, 2000, pp. 1–17.

[44] J. Denny and N. M. Amatoo, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Springer, 2013, pp. 297–312.

[45] J. Denny, K. Shi, and N. M. Amato, "Lazy toggle PRM: A single-query approach to motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2013, pp. 2407–2414.

[46] F. Ernst, L. Richter, L. Matthäus, V. Martens, R. Bruder, A. Schlaefer, and A. Schweikard, "Non-orthogonal tool/flange and robot/world calibration," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 8, no. 4, pp. 407–420, 2012.

[47] D. J. Evans, "On the representatation of orientation space," *Molecular Physics*, vol. 34, no. 2, pp. 317–325, 1977.

[48] A. Fabri and S. Pion, "CGAL: The computational geometry algorithms library," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2009, pp. 538–539.

[49] I. Fassi and G. Legnani, "Hand to sensor calibration: A geometrical interpretation of the matrix equation AX = XB," *Journal of Robotic Systems*, vol. 22, no. 9, pp. 497–506, 2005.

[50] E. Fogel and D. Halperin, "Exact and efficient construction of Minkowski sums of convex polyhedra with applications," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, Society for Industrial and Applied Mathematics, 2006, pp. 3–15.

[51] ——, "Exact and efficient construction of Minkowski sums of convex polyhedra with applications," *Computer-Aided Design*, vol. 39, no. 11, pp. 929–940, 2007.

[52] E. Fogel, D. Halperin, and C. Weibel, "On the exact maximum complexity of Minkowski sums of polytopes," *Discrete & Computational Geometry*, vol. 42, no. 4, p. 654, 2009.

[53] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[54] P. K. Ghosh, "A unified computational framework for Minkowski operations," *Computers & Graphics*, vol. 17, no. 4, pp. 357–378, 1993.

[55] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[56] A. Ginzburg, *Algebraic theory of automata*. New York, NY, USA: Academic Press, 1968.

[57] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 1996, pp. 171–180.

[58] Y. Gu, Y. He, K. Fatahalian, and G. Blelloch, "Efficient BVH construction via approximate agglomerative clustering," in *Proceedings of the 5th High-Performance Graphics Conference*, ACM, 2013, pp. 81–88.

[59] L. Guibas, L. Ramshaw, and J. Stolfi, "A kinetic framework for computational geometry," in *Foundations of Computer Science, 1983., 24th Annual Symposium on*, IEEE, 1983, pp. 100–111.

[60] P. Hachenberger, "Exact Minkowksi sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces," *Algorithmica*, vol. 55, no. 2, pp. 329–345, 2009.

[61] T. Hahn, Ed., *International tables for crystallography, volume A: space group symmetry*. International Union for Crystallography (IUCr), Chester, UK, 2002.

[62] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3-4, pp. 577–601, 2000.

[63] J. Hartmanis, *Algebraic structure theory of sequential machines*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1966.

[64] E. E. Hartquist, J. Menon, K. Suresh, H. B. Voelcker, and J. Zagajac, "A computing strategy for applications involving offsets, sweeps, and Minkowski operations," *Computer-Aided Design*, vol. 31, no. 3, pp. 175–183, 1999.

[65] T. C. Henderson, X. Fan, S. Devnani, S. Kumar, E. Cohen, and E. Grant, "Symmetry as an organizational principle in cognitive sensor networks," Technical Report UUCS-09-005, The University of Utah, Tech. Rep., 2009.

[66] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, vol. 3, 2003, pp. 4420–4426.

[67] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *IEEE International Conferencce on Robotics and Automation (ICRA)*, IEEE, vol. 3, 1997, pp. 2719–2726.

[68] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk, and D. Manocha, "V-COLLIDE: Accelerated collision detection for VRML," in *Proceedings of the second symposium on Virtual reality modeling language*, ACM, 1997, 117–ff.

[69] S. Iwata, Y. Nakatsukasa, and A. Takeda, "Computing the signed distance between overlapping ellipsoids," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2359–2384, 2015.

[70] R. Jackendoff, *Semantics and cognition*. Cambridge, MA, USA: The MIT Press, 1985.

[71] A. Jaklic, A. Leonardis, and F. Solina, *Segmentation and recovery of superquadrics*. Springer Science & Business Media, 2013, vol. 20.

[72] T. Janssen, *Crystallographic groups*. New York, NY, USA: Elsevier, 1973.

[73] X. Jia, Y.-K. Choi, B. Mourrain, and W. Wang, "An algebraic approach to continuous collision detection for ellipsoids," *Computer Aided Geometric Design*, vol. 28, no. 3, pp. 164–176, 2011.

[74] D. Jung and K. K. Gupta, "Octree-based hierarchical distance maps for collision detection," *Journal of Robotic Systems*, vol. 14, no. 11, pp. 789–806, 1997.

[75] M. Karnik, S. K. Gupta, and E. B. Magrab, "Geometric algorithms for containment analysis of rotational parts," *Computer-Aided Design*, vol. 37, no. 2, pp. 213–230, 2005.

[76] L. E. Kavraki, "Computation of configuration-space obstacles using the fast Fourier transform," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 408–413, 1995.

[77] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[78] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, Springer, 1986, pp. 396–404.

[79] P. Khosla and R. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 1988, pp. 1778–1784.

[80] C. Kiddon and P. Domingos, "Symmetry-based semantic parsing," in *NIPS Workshop on Learning Semantics*, 2015.

[81] J. J. Kuffner, "Effective sampling and distance metrics for 3D rigid body path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, vol. 4, 2004, pp. 3993–3998.

[82] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, vol. 2, 2000, pp. 995–1001.

[83] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal toolbox (ET)," in *IEEE Conference on Decision and Control (CDC)*, IEEE, 2006, pp. 1498–1503.

[84] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[85] E. E. Lattman, "Optimal sampling of the rotation function," *Acta Crystallographica Section B: Structural Crystallography and Crystal Chemistry*, vol. 28, no. 4, pp. 1065–1068, 1972.

[86] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Department, Iowa State University, Tech. Rep., 1998.

[87] A. Leonardis, A. Jaklic, and F. Solina, "Superquadrics for segmenting and modeling range data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1289–1295, 1997.

[88] A. Li, L. Wang, and D. Wu, "Simultaneous robot-world and hand-eye calibration using dual-quaternions and Kronecker product," *The International Journal of Physical Sciences*, vol. 5, no. 10, pp. 1530–1536, 2010.

[89] H. Li, Q. Ma, T. Wang, and G. Chirikjian, "Simultaneous hand-eye and robot-world calibration by solving the $AX = YB$ problem without correspondence," *IEEE Robotics and Automation Letters (RA-L)*, 2016.

[90] J.-M. Lien, "A simple method for computing Minkowski sum boundary in 3d using collision detection," in *Algorithmic foundation of robotics VIII*, Springer, 2009, pp. 401–415.

[91] ——, "Covering Minkowski sum boundary using points with applications," *Computer Aided Geometric Design*, vol. 25, no. 8, pp. 652–666, 2008.

[92] ——, "Hybrid motion planning using Minkowski sums," *Robotics: Science and Systems (RSS)*, 2008.

[93] ——, "Point-based Minkowski sum boundary," in *Pacific Conference on Computer Graphics and Applications*, IEEE, 2007, pp. 261–270.

[94] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 1991, pp. 1008–1014.

[95] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 2439–2446, 2018.

[96] Y. Liu and R. Collins, "A computational model for repeated pattern perception using frieze and wallpaper groups," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2000, pp. 537–544.

[97] D. S. Lopes, M. T. Silva, J. A. Ambrósio, and P. Flores, "A mathematical framework for rigid contact detection between quadric and superquadric surfaces," *Multibody System Dynamics*, vol. 24, no. 3, pp. 255–280, 2010.

[98] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. 100, no. 2, pp. 108–120, 1983.

[99] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[100] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[101] Q. Ma, Z. Goh, S. Ruan, and G. Chirikjian, "Probabilistic approaches to the AXB=YCZ calibration problem in multi-robot systems," *Autonomous Robots*, vol. 42, pp. 1497–1520, 2018.

[102] Q. Ma, H. Li, and G. Chirikjian, "New probabilistic approaches to the $AX = XB$ hand-eye calibration without correspondence," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016.

[103] Q. Ma and G. S. Chirikjian, "A closed-form lower bound on the allowable motion for an ellipsoidal body and environment," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE)*, ASME, 2015, V05CT08A055.

[104] A. Makhal, F. Thomas, and A. P. Gracia, "Grasping unknown objects in clutter by superquadric representation," in *IEEE International Conference on Robotic Computing (IRC)*, IEEE, 2018, pp. 292–299.

[105] J. Matousek, *Geometric discrepancy: An illustrated guide*. Springer Science & Business Media, 2009, vol. 18.

[106] M. Minsky and S. Papert, *Perceptrons: an introduction to computational geometry*. Cambridge, MA, USA: The MIT Press, 1987.

[107] A Morawiec and D. Field, "Rodrigues parameterization for orientation and misorientation distributions," *Philosophical Magazine A*, vol. 73, no. 4, pp. 1113–1130, 1996.

[108] K. Moustakas, G. Nikolakis, D. Koutsonanos, D. Tzovaras, and M. G. Strintzis, "Haptic feedback using an efficient superquadric based collision detection method," in *World Haptics Conference (WHC)*, IEEE, 2005, pp. 649–650.

[109] K. Moustakas, D. Tzovaras, and M. G. Strintzis, "SQ-Map: Efficient layered collision detection and haptic rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 80–93, 2007.

[110] A. Müller, "Screw and Lie group theory in multibody dynamics," *Multibody System Dynamics*, vol. 42, no. 2, pp. 219–248, 2018.

[111] R. M. Murray, S. S. Sastry, and Z. Li, *A Mathematical Introduction to Robotic Manipulation*, 1st. Boca Raton, FL, USA: CRC Press, Inc., 1994, ISBN: 0849379814.

[112] N. Najmaei and M. R. Kermani, "Superquadric obstacle modeling and a danger evaluation method with applications in safe planning for human-safe industrial robots," in *IEEE International Conference on Technologies for Practical Robot Applications*, IEEE, 2009, pp. 129–134.

[113] S. Nelaturi and V. Shapiro, "Configuration products and quotients in geometric modeling," *Computer-Aided Design*, vol. 43, no. 7, pp. 781–794, 2011.

[114] H. P. Nii, E. A. Feigenbaum, and J. J. Anton, "Signal-to-symbol transformation: HASP/SIAP case study," *AI magazine*, vol. 3, no. 2, pp. 23–23, 1982.

[115] M. R. Pac and D. O. Popa, "Interval analysis of kinematic errors in serial manipulators using product of exponentials formula," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 525–535, 2013.

[116] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2012, pp. 3859–3866.

[117] F. Park and B. Martin, "Robot sensor calibration: Solving AX = XB on the euclidean group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, 1994.

[118] D. Paschalidou, A. O. Ulusoy, and A. Geiger, "Superquadrics revisited: Learning 3D shape parsing beyond cuboids," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 344–10 353.

[119] M. Peternell and T. Steiner, "Minkowski sum boundary surfaces of 3D-objects," *Graphical Models*, vol. 69, no. 3-4, pp. 180–190, 2007.

[120] S. B. Pope, "Algorithms for ellipsoids," *Cornell University Report No. FDA-08-01*, 2008.

[121] R. Portal, J. Dias, and L. de Sousa, "Contact detection between convex superquadric surfaces," *Archive of Mechanical Engineering*, vol. 57, no. 2, pp. 165–186, 2010.

[122] E. Rimon and S. P. Boyd, "Obstacle collision detection using best ellipsoid fit," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 2, pp. 105–126, 1997.

[123] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2006, pp. 895–900.

[124] S. Ruan, J. Ding, Q. Ma, and G. S. Chirikjian, "The kinematics of containment for N-dimensional ellipsoids," *Journal of Mechanisms and Robotics*, vol. 11, no. 4, p. 041 005, 2019.

[125] S. Ruan, J. S. Kim, and G. S. Chirikjian, "Symmetrical rigid body parameterizations for humanoid robots," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2016, pp. 1655–1661.

[126] S. Ruan, Q. Ma, K. L. Poblete, Y. Yan, and G. S. Chirikjian, "Path planning for ellipsoidal robots and general obstacles via closed-form characterization of Minkowski operations," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Springer, 2018, pp. 3–18.

[127] S. Ruan, K. L. Poblete, Y. Li, Q. Lin, Q. Ma, and G. S. Chirikjian, "Efficient exact collision detection between ellipsoids and superquadrics via closed-form Minkowski sums," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 1765–1771.

[128] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 3rd. Upper Saddle River, NJ, USA: Pearson Education Inc., 2009.

[129] J.-R. Sack and J. Urrutia, *Handbook of computational geometry*. Elsevier, 1999.

[130] A Sanderson, "Parts entropy methods for robotic assembly system design," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, vol. 1, 1984, pp. 600–608.

[131] R. A. Sasongko, S. Rawikara, and H. J. Tampubolon, "UAV obstacle avoidance algorithm based on ellipsoid geometry," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 567–581, 2017.

[132] J. M. Selig, "Lie groups and Lie algebras in robotics," in *Computational Noncommutative Algebra and Applications*, Springer, 2004, pp. 101–125.

[133] K. Shi, J. Denny, and N. M. Amato, "Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 4659–4666.

[134] B. Shiffman, S. Lyu, and G. Chirikjian, "Mathematical aspects of molecular replacement. V. isolating feasible regions in motion spaces," *Acta Crystallographica Section A: Foundations and Advances*, 2020.

[135] Y. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX = XB," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 16–29, 1989.

[136] K. Shoemake, "Uniform random rotations," in *Graphics Gems III (IBM Version)*, Elsevier, 1992, pp. 124–132.

[137] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: The case for superquadrics with global deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 131–147, 1990.

[138] L. A. S. Sors and L. A. Santaló, *Integral geometry and geometric probability*. Cambridge university press, 2004.

[139] P Stein, "A note on the volume of a simplex," *The American Mathematical Monthly*, vol. 73, no. 3, pp. 299–301, 1966.

[140] H. T. Stokes and D. M. Hatch, *Isotropy subgroups of the 230 crystallographic space groups*. World Scientific, 1988.

[141] M. Studdert-Kennedy, "How did language go discrete," *Evolutionary prerequisites for language*, 2005.

[142] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[143] D. A. Suprunenko, *Matrix groups*, No. 45. American Mathematical Soc., 1976.

[144] M. J. Todd, *Minimum-volume ellipsoids: Theory and algorithms*. Society for Industrial and Applied Mathematics (SIAM), 2016, vol. 23.

[145] R. Tsai and R. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.

[146] G. Varadhan and D. Manocha, "Accurate Minkowski sum approximation of polyhedral models," in *Pacific Conference on Computer Graphics and Applications*, IEEE, 2004, pp. 392–401.

[147] ——, "Star-shaped Roadmaps-a deterministic sampling approach for complete motion planning," in *Robotics: Science and Systems (RSS)*, vol. 173, 2005.

[148] G. Vezzani, U. Pattacini, and L. Natale, "A grasping approach based on superquadric models," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1579–1586.

[149] J. Wang, L. Wu, M. Q.-H. Meng, and H. Ren, "Towards simultaneous coordinate calibrations for cooperative multiple robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2014, pp. 410–415.

[150] W. Wang, L. Zuo, and X. Xu, "A learning-based multi-RRT approach for robot path planning in narrow passages," *Journal of Intelligent & Robotic Systems*, vol. 90, no. 1-2, pp. 81–100, 2018.

[151] W. Wang, Y.-K. Choi, B. Chan, M.-S. Kim, and J. Wang, "Efficient collision detection for moving ellipsoids using separating planes," *Computing*, vol. 72, no. 1-2, pp. 235–246, 2004.

[152] W. Wang, J. Wang, and M.-S. Kim, "An algebraic condition for the separation of two ellipsoids," *Computer aided geometric design*, vol. 18, no. 6, pp. 531–539, 2001.

[153] Y. Wang and G. Chirikjian, "Nonparametric second-order theory of error propagation on motion groups," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1258–1273, 2008.

[154] C. Wellmann, C. Lillie, and P. Wriggers, "A contact detection algorithm for superellipsoids based on the common-normal concept," *Engineering Computations*, vol. 25, no. 5, pp. 432–442, 2008.

[155] T. Winograd and F. Flores, *Understanding computers and cognition: a new foundation for design*. Norwood, NJ, USA: Ablex Publishing Corporation, 1986.

[156] H. Wondratschek and U. Müller, Eds., *International tables for crystallography, volume A1: symmetry relations between space groups*. International Union for Crystallography (IUCr), Chester, UK, 2008.

[157] L. Wu, J. Wang, L. Qi, K. Wu, H. Ren, and M. Q. H. Meng, "Simultaneous hand-eye, tool-flange, and robot-robot calibration for comanipulation by solving the $AXB = YCZ$ problem," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 413–428, 2016.

[158] C. Wülker, S. Ruan, G. S. Chirikjian, *et al.*, "Quantizing Euclidean motions via double-coset decomposition," *Research*, vol. 2019, p. 1 608 396, 2019.

[159] S. Yan, S. Ong, and A. Nee, "Registration of a hybrid robot using the degradation-Kronecker method and a purely nonlinear method," *Robotica*, pp. 1–12, 2015.

[160] W. Yan, Z. Deng, J. Chen, H. Nie, and J. Zhang, "Precision grasp planning for multi-finger hand to grasp unknown objects," *Robotica*, vol. 37, no. 8, pp. 1415–1437, 2019.

[161] Y. Yan and G. Chirikjian, "Voronoi cells in Lie groups and coset decompositions: Implications for optimization, integration, and fourier analysis," in *IEEE Conference on Decision and Control (CDC)*, 2013, pp. 1137–1143.

[162] Y. Yan, "Geometric motion planning methods for robotics and biological crystallography," Ph.D. dissertation, Johns Hopkins University, 2014.

[163] Y. Yan and G. S. Chirikjian, "Almost-uniform sampling of rotations for conformational searches in robotics and structural biology," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2012, pp. 4254–4259.

[164] ——, "Closed-form characterization of the Minkowski sum and difference of two ellipsoids," *Geometriae Dedicata*, vol. 177, no. 1, pp. 103–128, 2015.

[165] Y. Yan, Q. Ma, and G. S. Chirikjian, "Path planning based on closed-form characterization of collision-free configuration-spaces for ellipsoidal bodies obstacles and environments," in *Proc. 1st Int. Workshop Robot Learn. Planning*, 2016, pp. 13–19.

[166] H.-Y. Yeh, S. Thomas, D. Eppstein, and N. M. Amato, "UOBPRM: A uniformly distributed obstacle-based PRM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2012, pp. 2655–2662.

[167] A. Yershova, S. Jain, S. M. Lavalle, and J. C. Mitchell, "Generating uniform incremental grids on SO(3) using the Hopf fibration," *The International journal of robotics research*, vol. 29, no. 7, pp. 801–812, 2010.

[168] H. Zhuang, Z. Roth, and R. Sudhakar, "Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form AX = YB," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 549–554, 1994.

Vita

Sipu Ruan obtained bachelor's degree in Mechanical Engineering at Harbin Institute of Technology, China in 2015. He then earned the Master's degree in Robotics at Johns Hopkins University, Baltimore, USA in 2017. He was admitted into the Mechanical Engineering PhD program at Johns Hopkins University in 2016.

Sipu Ruan received undergraduate honor award from Harbin Institute of Technology in 2015. He has served as research assistant since 2016 and assistant to the Editor-in-Chief of Robotica journal since 2018 at Johns Hopkins University.